Particle Swarm CMA-ES

Diploma Thesis
at the
Computational Biophysics Lab, ETH Zurich
in collaboration with the
Associate Institute for Signal Processing, TU Munich

Author: Benedikt Baumgartner, TUM, Mobility Student at ETH
Advisors: M.Sc. Christian L. Müller, Computational Biophysics Lab, ETH
          Prof. Dr. Ivo Sbalzarini, Computational Biophysics Lab, ETH
          Prof. Dr. Wolfgang Utschick, Institute for Signal Processing, TUM
Submission: June 16, 2008
Duration: December 17, 2007 - June 13, 2008
# Contents

Nomenclature

1 Introduction
  1.1 Natural Computation
  1.1.1 Evolutionary Computation (EC)
  1.1.2 Swarm Intelligence (SI)
  1.2 Motivation
  1.3 Goals
  1.4 Organization

2 Covariance Matrix Adaptation Evolution Strategy (CMA-ES)
  2.1 Introduction to CMA-ES
  2.1.1 The Evolution Strategy in CMA-ES
  2.2 Algorithm Description
    2.2.1 Sampling
    2.2.2 Selection and Recombination
    2.2.3 Adapting the Covariance Matrix
    2.2.4 Step-Size Control
    2.2.5 Implementational Concerns and Flow Chart
  2.3 Related Research
  2.4 Summary

3 Particle Swarm Optimization (PSO)
  3.1 Introduction to PSO
  3.2 Algorithm Description
  3.3 Related Research
  3.4 Summary

4 Particle Swarm CMA-ES (PS-CMA-ES)
  4.1 Introduction to PS-CMA-ES
  4.1.1 Parallel CMA-ES
  4.1.2 From Parallel CMA-ES to PS-CMA-ES
  4.2 Adapting the Covariance Matrix
    4.2.1 Eigenvector Rotation
  4.3 Biasing the Mean Value
  4.4 Strategy Work-Flow and Parameter Setting
    4.4.1 Information Exchange between CMA-ES Instances
    4.4.2 Frequency of PSO Update
    4.4.3 Weighting the Covariances
    4.4.4 Merging the Work Flows
Nomenclature

Abbreviations

**BC** Broadcast Channel

**BD** Block Diagonalization

**CMA-ES** Covariance Matrix Adaptation Evolution Strategy

**EC** Evolutionary Computation

**FES** Function Evaluations

**Grad-BD** Gradient BD

**Grad-MMSE** Gradient Minimum Mean Square Error

**IPOP-CMA-ES** Increasing Population Size CMA-ES

**LR-CMA-ES** Local Restart CMA-ES

**MIMO** Multiple-Input-Multiple-Output

**MPI** Message Passing Interface

**PCA** Principal Component Analysis

**PS-CMA-ES** Particle Swarm CMA-ES

**PSGES** Particle Swarm Guided Evolution Strategy

**PSO** Particle Swarm Optimization

**SI** Swarm Intelligence

**SQP** Sequential Quadratic Programming
1 Introduction

1.1 Natural Computation

The presented work is a new approach in the field of Natural Computing. In Computer Science, Natural Computing deals with the development of computational tools, inspired by natural systems. They are used to solve complex problems, mostly from the machine-learning and optimization domain. Apart from Neural Networks, Evolutionary Computation and Swarm Intelligence are two dominant techniques of nature-inspired computing and represent the frameworks of this thesis.

A new methodology, combining two well-known optimization strategies, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) and Particle Swarm Optimization (PSO), is presented and its performance is compared to similar optimization techniques. By combining the two algorithms, mutual interaction will hopefully lead to a powerful optimizer, emphasizing the strengths of each strategy while minimizing their shortcomings.

1.1.1 Evolutionary Computation (EC)

The roots of Evolutionary Computation (EC) can be traced back as far as the 1930s, although implementations of the basic idea were developed in the 1960s for the first time. Up until the early 1990s, three implementations of the same idea, namely Evolutionary Programming, Genetic Algorithm and Evolution Strategies developed separately, until they were considered as different flavours of one technology, Evolutionary Computing [1].

EC draws its inspiration from natural (Darwinian) evolution. Its applications are numerous and diverse. For example, EC methods were applied in communications and image processing. They were also used in finance and economics, as well as in transportation and logistics [2].

In this work, EC is used in the pure Computer Science/Engineering sense: evolutionary systems are seen as computational processes for solving complex problems. This is considered to be the fundamental metaphor of EC [1]. In order to use evolutionary systems as problem solvers, a characterization of a Darwinian evolutionary system is needed. De Jong [3] lists the features of a basic evolutionary process:

- one or more populations of individuals competing for limited resources,
- the notion of dynamically changing populations due to birth and death of individuals,
- a concept of fitness which reflects the ability of an individual to survive and reproduce, and
- a concept of variation inheritance: offspring closely resemble their parents, but are not identical.

Transferring this concept to an optimization strategy is intuitive. The environment in which the population lives can be seen as the problem space. Each individual represents
a candidate solution of the problem. The concept of fitness determines the quality of the solution, selecting individuals to survive and die. The variation inheritance produces new candidate solutions.

### 1.1.2 Swarm Intelligence (SI)

Another sub-category of Natural Computing is Swarm Intelligence (SI) [4]. Typically, SI-Systems are also made up of a population of candidate solutions, used to solve complex problems. Compared to EC, the main difference is, that the individuals of a population can communicate with each other. They exhibit collective behaviour, although there is no centralized form of controlling the individuals. Instead of selecting and reproducing new individuals as in EC, the concept of adaptation is different. Population members of a SI-based algorithm are able to move around in their environment. Two well-established algorithms using SI, are Ant Colony [5] and Particle Swarm Optimization [6].

### 1.2 Motivation

Parameter Optimization is a common problem in many fields of research. During the last years, Evolution Strategies, such as CMA-ES [7], and SI-based procedures, like PSO [8], became famous for solving complex optimization tasks [2, 9, 10, 11]. Both algorithms are appealing and well-established: CMA-ES because of its elaborate adaptation mechanism and PSO because of its simplicity. However, there are problems that can not be solved easily by any of the two algorithms, because of their search behaviours and methodologies. CMA-ES lacks the ability to integrate global information of the search space, PSO has difficulties in exploiting local information. In this work, a hybrid optimization procedure is developed. The new approach combines CMA-ES with concepts from PSO. Parallel CMA-ES runs shall be organized through Swarm Intelligence (PSO). Such a procedure will hopefully stress the advantages of CMA-ES and PSO, while some of their drawbacks are eliminated.

### 1.3 Goals

The objectives of this thesis are three-folded. First, a modular Fortran implementation of CMA-ES is designed. There exist implementations in common engineering programming languages like MATLAB, C++ and Java\(^1\), but not Fortran. Though Fortran is spuriously said to be a fairly old programming language, it has constantly been adapted in accordance with new programming techniques and hardware facilities, resulting in the latest Fortran-2003 standard [12]. It is exhaustively used by mathematicians and engineers and is a standard and efficient programming language for high performance computing [13]. It is optimized for numerical calculations and, together with conservative object-oriented aspects, a powerful language [14]. Therefore, a Fortran implementation is legitimate and useful.

\(^{1}\)available at [http://www.bionsik.tu-berlin.de/user/niko/cmaes_inmatlab.html](http://www.bionsik.tu-berlin.de/user/niko/cmaes_inmatlab.html)
Second, a new hybrid algorithm, combining CMA-ES and PSO, is developed. To successfully accomplish this challenging task, profound knowledge of CMA-ES and PSO is needed. Parallel code, that enables multiple CMA-ES instances to run at the same time, is required. Furthermore, a communication strategy between multiple CMA-ES runs, inspired by PSO, has to be developed. On an implementational level, the Message Passing Interface (MPI) is used as a communication framework. Tests on common benchmark functions will reveal the performance of the proposed optimization strategy in comparison to similar techniques.

To top off this thesis, the developed strategy is tested on a real-world optimization problem. Concerning the author’s background, the test case is chosen from the electrical engineering area and treats the rate balancing problem in the Multi-User Multiple-Input-Multiple-Output (MIMO) Broadcast Channel (BC) [15–17].

1.4 Organization

This work is structured as follows. The next chapter introduces CMA-ES, describing the core algorithm and enhancements. In chapter 3, the basic PSO algorithm is presented, highlighting its advantages and disadvantages. In chapter 4 the two algorithms are unified, resulting in a hybrid procedure, called Particle Swarm CMA-ES (PS-CMA-ES) in the following. The developed strategy is tested on common benchmark functions and its results and performance are analyzed in chapter 5. Next, the method is applied to the rate balancing problem in the MIMO-BC. Chapter 7 highlights the technical (implementational) details, before a summary and an outlook conclude this thesis.
2 Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

2.1 Introduction to CMA-ES

The CMA-ES is a stochastic, iterative method for real-valued-parameter optimization of non-linear, non-convex functions. It is a powerful optimization procedure and performs especially well in rugged search-landscapes with discontinuities, noise, local optima, etc. Learning the covariance matrix in CMA-ES is equal to learning a second-order model of the given objective function, analogous to the approximation of the inverse Hessian matrix in a Quasi-Newton method. Additionally, a so-called evolution path, containing information on the correlation between consecutive steps is recorded and used for the covariance matrix adaptation mechanism as well as for an auxiliary step-size control.

CMA-ES was introduced by N. Hansen A. Ostermeier in 1996 [7]. Major improvements of the very first idea made CMA-ES a highly elaborate optimization algorithm. In 2001, weighted recombination was introduced to CMA-ES [18]. Two years later, the so-called rank-µ-update greatly reduced time complexity [19]. Ongoing adjustments and modifications improved performance. It was found, that global search characteristics can be enhanced, if the population size is increased [20]. A very recent modification further reduces time and space complexity [21].

2.1.1 The Evolution Strategy in CMA-ES

Evolution Strategies [22, 23] aim at simulating Darwinian evolution and can be easily described with a flow chart (Figure 1). First of all, a population with a predefined number of individuals is initialized within the given problem space (mostly at random). The selection mechanism chooses individuals that will be considered as parents of the next generation. Based on a recombination mechanism, offspring from the given parent population will be created. In analogy to the concept of variation inheritance in EC, mutations ensure, that offspring resemble their parents, but are not identical. After creating the offspring, the selection operator will again choose individuals to be the parents of next generation’s offspring.

In 1975, H.-P. Schwefel [24] introduced the (µ+λ)-ES models, in which µ specifies the size of the parent population and λ the size of the offspring population. The most simple ES would be a (1+1)-ES, in which each population consists of only one individuum from which one offspring is created with a mutation operator.

Seeing evolution as an optimization process leaves room for interpretation and creativity, resulting in major questions:

- How can the selection operator be specified?
- What kind of recombination mechanism would be sensible?
- Is there a reasonable way of mutation?
2.2 Algorithm Description

In CMA-ES, elaborate ways to exploit the power of Evolution Strategies are incorporated. Starting from population size $\lambda$, $\mu$ individuals are selected based on their fitness. This is followed by an intermediate recombination. Based on the average position of the current population (size $\mu$), a single parent is created. Surrounding this parent, the next generation offspring is created, sampled from a multivariate normal distribution, that is determined by its mean and covariance.

In the next section the mathematical formulation of CMA-ES is given, followed by two variations of the strategy, the Local Restart CMA-ES (LR-CMA-ES) [25] and the Increasing Population Size CMA-ES (IPOP-CMA-ES) [26].

2.2 Algorithm Description

This section summarizes the tutorials on CMA-ES by N. Hansen [27, 28], which are highly recommendable along with [18] for more detailed and background information.

2.2.1 Sampling

In CMA-ES, the population offspring for the next generation ($g+1$) is generated by sampling a multivariate normal distribution with mean $\mathbf{m} \in \mathbb{R}^n$ and covariance $\mathbf{C} \in \mathbb{R}^{n \times n}$. A normal distribution has maximum entropy with finite variance, which relates to making the least possible assumptions on the objective function in the distribution.
shape. In addition, the sampling is controlled by the overall standard deviation, step-size $\sigma$. If $x_k^{(g)}$ denotes the $k$-th individual at generation $g$, the sampling reads

$$x_k^{(g+1)} \sim m^{(g)} + \sigma^{(g)} \mathcal{N}(0, C^{(g)}) \quad \text{for } k = 1, \ldots, \lambda. \quad (2.1)$$

Obvious questions are, how to calculate $m$, $C$ and $\sigma$ for the next generation ($g + 1$).

### 2.2.2 Selection and Recombination

From the $k$ sampled points, $\mu$ are selected and ranked in ascending order, according to their fitness value. The new mean value can be calculated as a weighted intermediate recombination of the selected points:

$$m^{(g+1)} = \sum_{i=1}^{\mu} w_i x_{\lambda;\lambda}^{(g+1)} \quad (2.2)$$

$$\sum_{i=1}^{\mu} w_i = 1, \quad w_1 \geq w_2 \geq \ldots \geq w_\mu > 0 \quad (2.3)$$

$w_i$ are positive weight coefficients for recombination and $x_{\lambda;\lambda}^{(g+1)}$ denotes the $i$-th ranked individual of the $\lambda$ sampling points $x_{\lambda}^{(g+1)}$. For example, selected sample points could be equally weighted ($w_i = \frac{1}{\mu}$). In the standard CMA-ES implementation, sample weights are decreased super-linearly: $w_i = \log(\frac{\lambda-1}{2} + 1) - \log(i)$.

### 2.2.3 Adapting the Covariance Matrix

Adapting the covariance matrix of the distribution is a complex step, that consists of three sub-procedures: the rank-$\mu$-update, the rank-one-update and cumulation. They are described in the following paragraphs and resemble a Principal Component Analysis (PCA) of steps, sequentially in time and space. The goal of the adaptation mechanism is to increase the probability of successful consecutive steps.

**Rank-$\mu$-Update.** The empirical covariance matrix is given as

$$C_{emp}^{(g+1)} = \frac{1}{\lambda - 1} \sum_{i=1}^{\lambda} \left( x_i^{(g+1)} - \frac{1}{\lambda} \sum_{j=1}^{\lambda} x_j^{(g+1)} \right) \left( x_i^{(g+1)} - \frac{1}{\lambda} \sum_{j=1}^{\lambda} x_j^{(g+1)} \right)^T. \quad (2.4)$$

In his work, Hansen uses a slightly modified formulation. Instead of taking the mean value of the actually realized samples for the covariance calculation, as Equation (2.4) suggests, the true mean value of the sampled distribution serves as the reference point:

$$C_{\lambda}^{(g+1)} = \frac{1}{\lambda} \sum_{i=1}^{\lambda} \left( x_i^{(g+1)} - m^{(g)} \right) \left( x_i^{(g+1)} - m^{(g)} \right)^T. \quad (2.5)$$
Using only selected samples and introducing weighting just as in Equation (2.2), the covariance matrix is calculated as

\[ C_{\mu}^{(g+1)} = \sum_{i=1}^{\mu} w_i \left( x_{i\lambda}^{(g+1)} - m^{(g)} \right) \left( x_{i\lambda}^{(g+1)} - m^{(g)} \right)^T \] (2.6)

and can be interpreted as an estimator for the distribution of selected steps. This means, that sampling from \( C_{\mu}^{(g+1)} \) tends to reproduce selected steps.

The rank-\( \mu \)-update uses \( \mu > 1 \) vectors to update the covariance matrix at each generation step. In addition, information from previous generations is used. More recent generations will have a stronger influence on the covariance matrix adaptation. This is realized through exponential smoothing. With learning rate \( c_{\text{cov}} \), the rank-\( \mu \)-update reads

\[ C^{(g+1)} = (1 - c_{\text{cov}}) C^{(g)} + c_{\text{cov}} \left( 1 - \frac{1}{\sigma^{(g)} \mu_{\text{eff}}} \right) \mu_{\text{eff}}^{(g+1)} - m^{(g)} \left( y^{(g+1)} \right)^T, \] (2.7)

where \( y^{(g+1)} = (x_{i\lambda}^{(g+1)} - m^{(g)}) / \sigma^{(g)} \).

**Rank-One-Update and Cumulation.** The rank-one-update uses only a single selected step for the covariance matrix adaptation. Together with the concept of cumulation, correlations between consecutive steps are exploited. Conceptually, the evolution path is the path that the strategy takes over a number of generations. It can be expressed as the sum of consecutive steps of the mean value \( m \) [27]. The recursive construction of the evolution path, \( p_c^{(g+1)} \in \mathbb{R}^n \), with \( p_c^{(0)} = 0 \), is referred to as cumulation:

\[ p_c^{(g+1)} = (1 - c_c) p_c^{(g)} + \sqrt{c_c(2 - c_c) \mu_{\text{eff}}} \frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}}, \] (2.8)

where \( c_c \leq 1 \) is the backward time horizon and \( \mu_{\text{eff}} \) a measure, paraphrased as variance effective selection mass:

\[ \mu_{\text{eff}} = \left( \sum_{i=1}^{\mu} w_i^2 \right)^{-1}. \] (2.9)

**Combined Adaptation.** Combining the rank-\( \mu \)-update and cumulation, the final covariance matrix update rule can be expressed as

\[ C^{(g+1)} = (1 - c_{\text{cov}}) C^{(g)} + \frac{c_{\text{cov}}}{\mu_{\text{cov}}} \underbrace{p_c^{(g+1)} p_c^{(g+1)^T}}_{\text{rank-one-update}} + c_{\text{cov}} \left( 1 - \frac{1}{\mu_{\text{cov}}} \right) \] \times \sum_{i=1}^{\mu} w_i y_{i\lambda}^{(g+1)} \left( y_{i\lambda}^{(g+1)} \right)^T, \] (2.10)
2.2 Algorithm Description

2.2.4 Step-Size Control

In addition to the covariance matrix adaptation rule, a step-size control is introduced, that adapts the overall scale of the distribution based on information obtained by the evolution path. The following rationale is applied: If the evolution path is long and single steps are pointing more or less to the same direction, the step-size should be increased. On the other hand, if the evolution path is short and single steps cancel each other out, the step-size should be decreased.

Similar to Equation (2.8), the step-size evolution path $p_\sigma$ is initialized with $p_\sigma^{(0)} = 0$ in generation 0 and in the subsequent generations calculated as

$$p_\sigma^{(g+1)} = (1 - c_\sigma)p_\sigma^{(g)} + \sqrt{c_\sigma(2 - c_\sigma) \mu_{\text{eff}} \mathbf{C}^{(g)}^{-\frac{1}{2}} \frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}}},$$

(2.11)

with $c_\sigma < 1$ again being the backward time horizon of the evolution path. When $c_\sigma = 1$, only the most recent step contributes to the cumulation. $m^{(g+1)} - m^{(g)}$ gives the current step and $\sqrt{c_\sigma(2 - c_\sigma) \mu_{\text{eff}}}$ is a normalization constant. The main difference to Equation (2.8) is the term $\mathbf{C}^{(g)}^{-\frac{1}{2}}$, representing a transformation, which makes the expected length of $p_\sigma^{(g+1)}$ independent of its direction.\footnote{2}$\mathbf{C}^{(g)}^{-\frac{1}{2}} = \mathbf{B}^{(g)}\mathbf{D}^{(g)-1}\mathbf{B}^{(g)^T}$ (see Appendix A) and scales the step $m^{(g+1)} - m^{(g)}$ within the coordinate system given by $\mathbf{B}^{(g)}$.

Now a step-size adaptation rule can be formulated. Hansen reflects, that selection ideally does not bias the length of the evolution path, $||p_\sigma^{(g+1)}||$, and that the length is equal to its expected length under random selection, which is simply equal to the expected length of a random normal vector, $E||\mathcal{N}(0, \mathbf{I})||$.

Comparing $||p_\sigma^{(g+1)}||$ and $E||\mathcal{N}(0, \mathbf{I})||$ results in the final step-size adaptation rule\footnote{Please refer to [28] for a detailed derivation of $\sigma^{(g+1)}$.}:

$$\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma} \left(\frac{||p_\sigma^{(g+1)}||}{E||\mathcal{N}(0, \mathbf{I})||} - 1\right)\right).$$

(2.12)

2.2.5 Implementational Concerns and Flow Chart

Parameters. An uprising question is, how to set the strategy parameters $\lambda$, $\mu$, $w_i$, $c_\sigma$, $d_\sigma$, $c_c$, $\mu_{\text{cov}}$ and $c_{\text{cov}}$. In [28], default values are given, chosen and tested to be applicable to a wide range of optimization problems and therefore considered as a robust setting. A discussion on the parameter setting can be found in [18]. The population size, for example, is by default set to $4 + \lceil 3 \ln (n) \rceil$. Apart from the problem formulation and optional parameters, two strategy parameters can be chosen by the user: the initial step-size and the starting point. It is proposed, to use an initial step-size of 30% of the constrained region in search space. The starting position is drawn from a random uniform distribution within the problem space [28].
2.3 Related Research

Stop Criteria. In order to terminate an evolutionary algorithm, stop criteria have to be defined. For example, the algorithm should stop, if a given number of function evaluations or a given level of accuracy is reached. In the standard CMA-ES algorithm several stop criteria are specified and given in [28].

Boundary Handling. Many optimization problems are constrained and box boundaries are sometimes needed to limit down search space. In the CMA-ES, a rather simple way of handling such constraints is pursued: samples without the feasible region are re-sampled until they are in the feasible region again. The handling of boundaries and constraints in CMA-ES is still an ongoing research topic [28].

Flow Chart. A flow chart of CMA-ES, summarizing the main steps, is given in Figure 2. Later on, when introducing PS-CMA-ES, the chart will serve as a reference to explain the workflow of the approach.

2.3 Related Research

The CMA-ES was originally introduced to enhance local search properties of evolution strategies. In 2005 Auger and Hansen [25] introduced a local restart version of CMA-ES, referred to as LR-CMA-ES in order to stress local search characteristics even more. An initial step-size hundred times smaller than the recommended default values was used and a restart of the procedure was launched whenever specific stop criteria were met. Choosing such a small step-size, one might intuitively assume, that the strategy resembles a pure local search. But still, the step-size adaptation allows to perform global search as well.

In the same year, the authors presented a CMA-ES procedure with successively increasing population size, called IPOP-CMA-ES [26]. At every restart, the population size is increased by a factor of two.

Both strategies have been benchmarked as proposed in [29] and revealed similar performance on unimodal test functions, while IPOP-CMA-ES clearly outperformed LR-CMA-ES on most of the tested multimodal functions. Hansen and Kern assert that IPOP-CMA-ES performs well on multimodal functions where there is an underlying global topology [20] as CMA-ES is able to detect that structure.

Discussing this effect, Lunacek and Whitley [30] introduced a metric, called dispersion, that quantifies the proximity of the best regions in the search space. Using this metric, it can be predicted how efficient and effective CMA-ES will be on certain objective functions.

Another contribution to CMA-ES strategies was made by Jastrebski and Arnold [31]. Their so-called Active-CMA-ES uses information about unsuccessful offspring candidate solutions in order to actively reduce variances of the mutation distribution in unpromising directions of the search space.

The CMA-ES was applied to numerous problems from different areas such as biology, computer science, optics communications and many more. A list of more than 100 applications referring to CMA-ES can be found at http://www.bionik.tuberlin.de/user/niko/cmaapplications.pdf.
2.3 Related Research

Initialization: define problem space, set starting position, set initial $\sigma$, optional settings

Create sample points

Sort by fitness and compute weighted mean

Update evolution paths

Update covariance matrix

Update step-size $\sigma$

Error management

Selection

Recombination

Criteria ?

yes

STOP

no

Figure 2: CMA-ES Generation Loop Flow Chart
2.4 Summary

This chapter introduced the CMA-ES, an iterative optimization procedure, originally designed to enhance local search. It is applicable to non-linear, non-convex optimization problems in continuous domains. Usually it is applied if gradient-based methods fail due to a rugged search space. CMA-ES is an evolutionary algorithm and approximates a second-order model of the objective function. Two principles dominate the characteristics of CMA-ES. First, a PCA of successful mutation steps is conducted, increasing the likelihood for successful steps to appear again in following iterations. Second, the concept of cumulation is exploited, controlling the mutation strength of generated samples and preventing the population from premature convergence.

It was empirically shown, that restarts with increasing population size improve the global search performance and help to avoid local optima, while small population sizes usually lead to faster convergence. On multimodal functions with an underlying global structure, e.g. the Rastrigin function, CMA-ES with large population size is able to learn the global topology, while it has difficulties on multimodal functions with no such layout (e.g. the Schwefel function).

Very appealing characteristics of CMA-ES are its invariance to order preserving transformations of the objective function value and to angle preserving transformations of the search space (rotation, reflection and translation) [32]. Yet another convenient attribute is the self-adaptation of strategy internal parameters. This makes it an as is useable optimization method, applicable to a large amount of problems and is probably one of the reasons of its great success in various research areas.
3 Particle Swarm Optimization (PSO)

3.1 Introduction to PSO

Particle Swarm Optimization (PSO) was introduced by Kennedy and Eberhart [8] in 1995 as a concept for the optimization of continuous nonlinear functions. It was discovered through simulation of simplified social models and has both roots in artificial life and evolutionary computation. Because of its appealing simplicity and computational inexpensiveness, PSO became an attractive and competitive population-based optimization algorithm. In recent years it has been applied to all kinds of optimization problems [11, 33] and much research has been done in this field, resulting in many different PSO strategies. The fundamental hypothesis of PSO, however, remained unchanged:

"...social sharing of information among conspeciates offers an evolutionary advantage." [8]

To a great extend PSO is inspired by nature and can be nicely described with an analogy. Consider a shoal looking for food. Each fish randomly swims around, hoping to get something to eat. It will remember places where it has previously encountered food, so it is drawn towards places with a high probability of success. But somehow it also knows about the place with the highest food density found so far by any of the other swarm members. So the fish is drawn towards that direction as well. Occasionally, a fish finds a location with an even bigger food density. Quickly, the whole shoal will be drawn to that location. With such a behaviour, the shoal explores the territory. Fishes fly over locations with high food density, constantly checking if they have found a better place before. Generally spoken, individuals gather and communicate. By sharing information, each individual increases its chance of survival. From a technical point of view, PSO is based on the communication among swarm members (particles) in order to find an optimal solution for a given optimization problem.

On a meta-level, information flow between individuals contributes to a paradigm referred to as Swarm Intelligence. Following a definition of Millonas [34], Swarm Intelligence is based on five principles:

1. **Proximity Principle:** The population should be able to carry out simple space and time computations.

2. **Quality Principle:** The population should be able to respond to quality factors in the environment.

3. **Principle of Diverse Response:** The population should not commit its activities along excessively narrow channels.

4. **Principle of Stability:** The population should not change its mode of behaviour every time the environment changes.

5. **Principle of Adaptability:** The population must be able to change behaviour mode when it is worth the computational price.
3.2 Algorithm Description

The Particle Swarm Optimizer basically preserves these principles and the idea of Swarm Intelligence is utilized, when PSO is merged with CMA-ES in the next chapter. Going back to the example, each fish is aware of its current position and is able to calculate the food density at that position, as the Proximity Principle claims. Second, the fishes are able to classify the food density at the current position into "low", "medium" and "high" food density, for example, and are able to balance it with the overall best location, preserving the Quality Principle. As the movements of each swarm member are influenced by both its own memory and a global memory, a diversity of response can be guaranteed. Going along with the Stability Principle, the population changes its mode of behaviour only when a location with a better density of food was found, and, adhering to the Principle of Adaptability, it does change its behaviour, when a more attractive location was found.

In the next section the mathematical formulation of PSO is given, followed by a short review of existing PSO algorithms and their applications as well as a short summary, examining advantages and disadvantages of PSO.

3.2 Algorithm Description

A particle can be described by its position \( p \) and velocity \( v \) in a multi-dimensional, problem-dependent search space (\( \mathbb{R}^n \)). While moving through the problem space, particles evaluate the given fitness (cost) function and update their velocity via an update rule that not only follows local, but also incorporates global information. Each particle keeps track of the best solution it has found so far (the position with the best fitness function value), called local best and in the following denoted as \( p_{l,best} \). And it is also aware of the current global best position, detected by any swarm member (\( p_{g,best} \)). If \( p(t) \) denotes the particle position at time \( t \), the velocity update and the position after the update can be described by the following equations:

\[
\begin{align*}
v(t+1) &= v(t) + c_1 \cdot r_1 \cdot (p_{l,best}(t) - p(t)) + \\
& \quad c_2 \cdot r_2 \cdot (p_{g,best}(t) - p(t)) \\
p(t+1) &= p(t) + v(t+1)
\end{align*}
\]

Equation 3.1 has two main components: One is the attraction towards the particle’s local best position (line 2), and the other one the attraction towards the global best position (line 3). \( r_1 \) and \( r_2 \) are uniform random numbers in the interval [0,1], introducing a stochastic factor to the algorithm to simulate an unpredictable component of natural swarm behaviour. \( c_1 \) and \( c_2 \) are constants, defining how much local or global information influences the particles’ movement and are usually chosen \( c_1 = c_2 = 2 \). This makes the mean of the stochastic factor equal 1. In other words, the particles "overfly" the target about half the time [8]. Because \( c_1 \) pulls the particle towards its local best position, this factor is often called "cognitive rate" and \( c_2 \), as it draws the particle towards the global best position found by any swarm member, is called the
"social rate" factor. Whenever a particle detects a position better than the current global best, it updates $p_{g,best}$ and broadcasts it to the other particles in the swarm. This is done until any stop criterion is met. A Flow Chart of PSO is given in Figure 3.

![PSO Flow Chart](image)

Figure 3: PSO Flow Chart

### 3.3 Related Research

Particle Swarm Optimization led to a whole new research area. In recent years, many variations of the original algorithm have been developed and numerous applications from different areas were presented. For example, PSO was used to train neural networks [35–37], and Sahin et al. used PSO together with Bayesian Networks for fault
diagnosis of airplane engines [38]. Robinson et al. introduced PSO to the electromagnetics community and used it to optimize a corrugated horn antenna [39, 40], while Hung et al. used it for blind detection in frequency domain for a MC-CDMA communication system [41]. It has also been applied in a Polarisation Mode Dispersion (PMD) compensation algorithm for optical communication systems [42]. Alba et al. used a customized PSO algorithm for the Mobile Location Management (MLM) Problem in GSM networks [43].

Researchers also developed more enhanced PSO techniques to overcome problems of the original PSO algorithm. In 2002 Riget and Vesterstrøm presented a ”Diversity-Guided Particle Swarm Optimizer – the Attractive and Repulsive PSO (ARPSO)” [44]. Their algorithm alternates between exploring and exploiting behaviour based on a diversity measure, trying to counteract low diversity with fitness stagnation. If the measured diversity is larger than a threshold, particles will attract each other. If diversity decreases, particles will start to repel each other. Another and very recent variation was presented by Yisu et al., called the ”Landscape Adaptive Particle Swarm Optimizer (LAPSO)” [45]. They try to achieve a more robust and effective search by combining two sub-strategies. The first one introduces a distribution vector, taking the distribution of the whole swarm into account, which is used for velocity update. The second sub-strategy introduces a ”crossing-over” operator to escape from local minima.

### 3.4 Summary

PSO is a conceptually very simple algorithm and, because it only uses primitive mathematical operators, it is computationally simple and can be implemented with a few lines of code. Therefore it quickly became a popular optimization technique. Still, PSO in its original form has limitations. A well-known and major problem is premature convergence. As particles always move towards the global best, they easily cluster and diversity decreases, making it hard to escape local optima. Especially on highly multimodal functions this can be a real drawback, if particles were misled and cluster around a local minima, leading to a loss of exploration power of the swarm and fitness stagnation. As stated in [44], the fast information flow between particles seems to be the reason for clustering of particles. During the last decade, many advanced versions of PSO have been developed to overcome these limitations.
4 Particle Swarm Covariance Matrix Adaptation Evolution Strategy (PS-CMA-ES)

4.1 Introduction to PS-CMA-ES

We have shown in the previous chapters, that Particle Swarm Optimization (PSO) and Evolution Strategies are very popular and mighty algorithms for real–valued–parameter optimization problems. While PSO can be regarded as a global search algorithm, CMA-ES exhibits its strength at exploiting local structure. In this thesis, a new hybrid optimization algorithm is proposed. It combines PSO and CMA-ES in order to take advantage of their strengths and to overcome some of their weaknesses. CMA-ES is likely to get stuck in local optima on multi-modal problems and the coverage of the search space is usually sparse. We expect to outperform the standard CMA-ES on multi-modal functions without a global structure. Merging CMA-ES and PSO hopefully helps to overcome local optima. Furthermore, we expect, that performance on unimodal functions is at least equal to CMA-ES.

In the following sections we derive how to integrate PSO into the CMA-ES. The performance of the proposed algorithm is tested on common benchmark functions [29] and compared to similar techniques.

4.1.1 Parallel CMA-ES

In classical Particle Swarm Optimization, a swarm consists of a determined number of particles, moving around in search space, influenced by their own memory and global information, obtained through communication. In CMA-ES, selection and recombination are the leading operations.

First, we consider a swarm without any communication between swarm members. One particle is considered as a complete CMA-ES instance. Instead of sequentially running CMA-ES with different starting positions, it is now executed in parallel. The different CMA-ES runs are initialized at different positions in the search space (uniformly distributed) and exhibit equivalent behaviour in comparison to multiple executions of CMA-ES in sequential order. Parallel CMA-ES has been tested on standard benchmark functions, such as the Sphere function or the Rastrigin function. It has been compared to the current Matlab implementation of CMA-ES. The same random generator was used for sampling. The visualization of sample points and the covariance matrix, as well as numerical results confirmed the parity of parallel and sequential CMA-ES. The benchmark of independent, parallel CMA runs can serve as a baseline reference for PS-CMA-ES.

4.1.2 From Parallel CMA-ES to PS-CMA-ES

In 2007, Hsieh et al. [46] proposed a Particle Swarm Guided Evolution Strategy (PSGES) in which ES is considered a local and PSO a global search facility. They did pioneer work in combining the two search mechanisms and in creating a framework and work-flow for integrating SI into ES. The mutation ellipses of ES (similar to the covariance matrix in CMA-ES) were rotated based on the concept and mechanism of
4.2 Adapting the Covariance Matrix

SI via a novel mutation operator called *guided mutation*. Empirical results showed, that IPOP-CMA-ES outperforms PSGES and that LR-CMA-ES performs better on unimodal functions, while PSGES did a better job on multimodal test functions. The authors speculated, that PSGES might have a good global search mechanism but might need an enhanced local search operator.

As stated before, local search in CMA-ES is exceedingly sophisticated. Therefore it is an obvious suggestion to replace ES by CMA-ES to enhance local search performance. Having a validated parallelized implementation of CMA-ES available opens up the way towards a particle swarm guided CMA-ES. Still, as CMA-ES is a highly elaborate algorithm, thorough investigations are needed to transform CMA-ES into a SI-incorporating evolutionary algorithm. The hypothesis is, that exchanging information between parallel CMA-ES runs (organized by SI) enhances the performance of the pure CMA-ES algorithm. It is expected, that the convergence speed is lower than in CMA-ES, but that the proposed hybrid algorithm is superior on multimodal functions. The next sections deal with the incorporation of PSO concepts to CMA-ES. Similar to the guided mutation operator in PSGES, the covariance adaptation mechanism of CMA-ES is enhanced by global swarm information. Inspired by the attraction operators of PSO, the evolution path is modified by biasing the samples’ mean value towards the global best position.

4.2 Adapting the Covariance Matrix

In order to integrate the concept of PSO into CMA-ES, the matrix adaptation has to be altered slightly. The update rule should not only change the distribution such that it is likely to sample good candidates, but should also enable CMA-ES to generate solutions considering the current global best position in the swarm. This can be achieved by coupling the original covariance adaptation rule (Equation (2.10)) with a covariance matrix influenced by global information, making the updated covariance matrix \( C \) at generation \( g + 1 \) look like

\[
C^{(g+1)} = c \cdot C_{CMA}^{(g+1)} + (1 - c) \cdot C_{PSO}^{(g+1)}, \quad c \in [0, 1].
\]  

(4.1)

\( C_{CMA}^{(g+1)} \) represents the original adaptation rule as given in Equation (2.10). The second term of the equation introduces a covariance matrix \( C_{PSO}^{(g+1)} \). \( c \) is a scalar and can be considered as a weighting factor, defining to what extent global information should be taken into account when adapting the covariance matrix. If \( c = 1 \), the covariance matrix adaptation, as proposed in the standard CMA-ES algorithm, is not changed. If \( c = 0 \), only the covariance based on global information is considered during the update.

Similar to [46], \( C_{PSO}^{(g+1)} \) represents a rotated version of the covariance matrix \( C_{CMA}^{(g+1)} \). \( C_{CMA}^{(g+1)} \) is rotated, such that its principal eigenvector points to the global best position \( p_g \in \mathbb{R}^n \). This is done in order to redirect the evolution of CMA-ES towards the global best position and to increase the probability of samples resembling the global best solution. Figure 4 illustrates this procedure.

As \( C_{CMA}^{(g+1)} \) can be decomposed as \( C_{CMA}^{(g+1)} = BD^2B^T \) (see Appendix A), the rotated
covariance matrix can be constructed by rotating the eigenvectors of $C^{(g+1)}_{CMA}$ as described in the following section. Assuming that $B_{rot}^{(g)} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix containing the rotated eigenvectors of the current generation and $D^{(g)} \in \mathbb{R}^{n \times n}$ a diagonal matrix with the related eigenvalues, $C^{(g+1)}_{PSO}$ can be calculated as

$$C^{(g+1)}_{PSO} = B_{rot}^{(g)} \cdot (D^{(g)})^2 \cdot (B_{rot}^{(g)})^T.$$  \hspace{1cm} (4.2)

In the following it will be derived how to create $B_{rot}$.

$$\bullet \ p_g$$

![Figure 4: Rotation of $C^{(g+1)}_{CMA}$](image)

### 4.2.1 Eigenvector Rotation

If a rotation matrix $R \in \mathbb{R}^{n \times n}$ can be found, which rotates the principal eigenvector of $C$, $b_{main} \in \mathbb{R}^n$, such that it points to the global best position $p_g$, all other eigenvectors $b_i=1,...,n$ can be rotated, giving $B_{rot}$.

$$R \cdot b_{main} = a \cdot p_{best} \hspace{1cm} (4.3)$$

$$b_{i,rot} = R \cdot b_i \hspace{1cm} (4.4)$$

$$B_{rot} = [b_{1,rot}, b_{2,rot}, \ldots, b_{n,rot}]. \hspace{1cm} (4.5)$$

In Eq. (4.3), $a$ is a scalar and $p_{best} = p_g - m$ a vector, pointing from the current mean towards the global best position. Rotating $b_{main}$ onto $p_{best} \in \mathbb{R}^n$ is equivalent to making the cross product of the two vectors become 0:

$$(R \cdot b_{main}) \times (a \cdot p_{best}) = 0. \hspace{1cm} (4.6)$$

The problem of efficiently constructing the rotation matrix $R$ is also approached in [46]. For n-dimensional problems the task is subdivided into multiple rotations, each done on a two-dimensional plane. If the vector $b_{main}$ is rotated onto the vector $p_{best}$, there exist $\frac{n(n-1)}{2}$ possible pairs of axis and just as many rotation planes. The angle $\alpha$ between two vectors $x$ and $y$ is given by

$$\alpha = \cos^{-1} \left( \frac{x \cdot y}{||x|| \cdot ||y||} \right). \hspace{1cm} (4.7)$$

A counterclockwise rotation on a two-dimensional plane can be represented by a $2 \times 2$ rotation matrix:

$$R_{plane} = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix}. \hspace{1cm} (4.8)$$
Unfortunately, constructing the rotation matrix as shown in the preceding equations, is not a generally applicable procedure. Sometimes it might be necessary to rotate the vector just in the opposite direction, as illustrated in Figure 5. Then, the rotation matrix would look like

\[
R_{\text{plane}} = \begin{pmatrix}
\cos \alpha & \sin \alpha \\
-\sin \alpha & \cos \alpha
\end{pmatrix}.
\]  

(4.9)

![Diagram of rotation matrix](image)

Figure 5: Depending on the rotation direction, \(R_{\text{plane}}\) looks different.

Therefore, constructing the matrix as described above is not convenient, because it is crucial to have a generally applicable rotation procedure.

Instead, *Givens Rotations* [47] are used, represented by the rotation matrix \(G \in \mathbb{R}^{2\times2}\), which describes a unique rotation of a vector onto one axis. Let us assume the 2–dimensional case (\(\mathbb{R}^2\)). We rename the vectors \(\mathbf{b}_{\text{main}}\) and \(\mathbf{p}_{\text{best}}\) into \(\mathbf{b}_{\text{plane}}\) and \(\mathbf{p}_{\text{plane}}\). Performing a Givens rotation on \(\mathbf{b}_{\text{plane}}\) results in

\[
G_b \cdot \mathbf{b}_{\text{plane}} = \begin{pmatrix} v_1 \\ 0 \end{pmatrix},
\]  

(4.10)

while the Givens rotation of \(\mathbf{p}_{\text{plane}}\) gives

\[
G_p \cdot \mathbf{p}_{\text{plane}} = \begin{pmatrix} w_1 \\ 0 \end{pmatrix}.
\]  

(4.11)

Now \(\mathbf{b}_{\text{plane}}\) can consecutively be mapped onto \(\mathbf{p}_{\text{plane}}\), as depicted in Figure 6: First, both vectors are rotated onto one axis, giving \(G_b \) and \(G_p\). If \(G_p^{-1}\) is applied to the rotated vector \(\mathbf{b}_{\text{plane}}\), denoted as \(\mathbf{v}\) in Figure 6(a), the resulting vector will be \(\mathbf{b}_{\text{plane,rot}}\):

\[
\mathbf{b}_{\text{plane,rot}} = G_p^{-1} \cdot \mathbf{v} = G_p^{-1} \cdot G_b \cdot \mathbf{b}_{\text{plane}}
\]  

(4.12)

As Givens rotations are orthogonal [47], the inverse of \(G_p\) is substituted by its transpose \((G_p^{-1} = G_p^T)\) to ease calculations. The overall plane rotation matrix \(R_{\text{plane}}\) is obvious:

\[
R_{\text{plane}} = G_p^T \cdot G_b
\]  

(4.13)
4.2 Adapting the Covariance Matrix

\[ b_{\text{plane}} = G_b \cdot b_{\text{plane}} \]

(a) Givens rotation of \( b_{\text{plane}} \)

\[ w = G_p \cdot p_{\text{plane}} \]

(b) Givens rotation of \( p_{\text{plane}} \)

\[ b_{\text{plane, rot}} = G_p^{-1} \cdot v \]

(c) Final rotation

Figure 6: Two consecutive Givens Rotations are applied to rotate \( b_{\text{plane}} \) onto \( p_{\text{plane}} \).

In order to extend the rotation procedure to more than two dimensions, rotations, as described above, need to be conducted for all possible pairs of axes. As described in [48], a \( n \times n \) rotation matrix can be decomposed as a product of elementary rotation matrices:

\[
R_{(n \times n)} = \prod_{i=1}^{n-1} \prod_{j=i+1}^{n} R_{ij} \tag{4.14}
\]

\( R_{ij} \) is a \( n \times n \) matrix and describes the plane rotation of the pairs of axes \( (i, j) \). It can be considered as a rank-two correction to the identity:

\[
R_{ij} \cdot R_{\text{plane}} = \begin{pmatrix}
1 & \ldots & 0 & \ldots & 0 & \ldots & 0 \\
\vdots & \ddots & \vdots & \ldots & \vdots & \ddots & \vdots \\
0 & \ldots & R_{\text{plane}}(1, 1) & \ldots & R_{\text{plane}}(1, 2) & \ldots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots & \ddots & \vdots \\
0 & \ldots & R_{\text{plane}}(2, 1) & \ldots & R_{\text{plane}}(2, 2) & \ldots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
0 & \ldots & 0 & \ldots & 0 & \ldots & 1 \\
1 & i & j & \ldots & 1 \\
\end{pmatrix} \tag{4.15}
\]

The \( n \)-dimensional vectors \( b_{\text{main}} \) and \( p_{\text{best}} \) are mapped to the 2-D vectors \( b_{\text{plane}} \) and \( p_{\text{plane}} \):

\[
b_{\text{plane}} \leftarrow b_{\text{main}}(i, j) \tag{4.16}
\]

\[
p_{\text{plane}} \leftarrow p_{\text{best}}(i, j) \tag{4.17}
\]

Via Equations (4.15) – (4.17), rotation matrices \( R_b \) and \( R_p \) can be constructed for both \( b_{\text{main}} \) and \( p_{\text{best}} \). The overall rotation matrix is

\[
R = R_p^T \cdot R_b \tag{4.18}
\]

The summary of how to create \( R \), such that Equation (4.3) holds, is given in Algorithm 1 in Pseudo-Code notation.
4.3 Biasing the Mean Value

Preliminary tests showed, that only diversifying the covariance matrix update rule is not sufficient to gain pleasing results. Visualizing the evolution of this first approach, displayed an apparent behaviour pattern. Despite the rotation of the covariance matrix, CMA-ES runs that got stuck in local optima of multimodal functions (e.g. on the Schwefel function) were not able to escape. Reviewing the basic concepts of PSO, which was intended to cope with just that difficulty, resulted in introducing a second operation driven by SI. Not only are the covariance matrices rotated but also a bias is added to the mean value \( m^{(g)} \). This bias will help CMA-ES runs to get away from local optima and is alike the velocity update rule of PSO, dragging the run towards the swarm optimum \( p_g \). The bias has a strong impact on CMA-ES and must therefore be chosen very carefully.

In the style of Equation (3.2) the mean value is updated via

\[
m^{(g+1)} \leftarrow m^{(g+1)} + \text{bias}.
\]  

(4.19)

Note that the bias is added \textbf{after} the recombination step. It is not added to the bias of the current generation \( m^{(g)} \), but to the updated mean value of generation \( g + 1 \).

Choosing a bias, such that performance enhancement on a wide range of test functions can be observed, is a very demanding mission, already indicated by the numerous variations of the PSO velocity update rule that exist in current research.

Biasing \( m^{(g+1)} \) requires to differentiate between several scenarios. First, the CMA-ES instance that produced \( p_g \) does not need to be biased, as it is already located in the
near surrounding of $p_g$. For all other swarm members it is distinguished between runs, that are converging, and therefore considered as falling into a local optimum, and runs that are still exploring. First tries revealed, that runs trapped in local optima need a much stronger bias than runs in exploration mode. Small biases would make them fall into the same optima again within very few subsequent generations. Assigning a large bias to runs that are still exploring is also not convenient, because it would lead to clustering at $p_g$ and exploration power would be lost. A first, very simple metric to decide whether a CMA-ES run is converging or exploring, is the step-size $\sigma$.

Comparing $\sigma$ to the distance from a CMA-ES instance to the global best position, given by $p_{best} = p_g - m$, yields two bias update rules. If $\sigma$ is much smaller than that distance, the run is supposed to be trapped and is biased by half the length of $p_{best}$:

$$bias_1 = 0.5p_{best}$$

(4.20)

Otherwise, a smaller bias is preferable, to avoid premature convergence, and is chosen such that runs are drawn with a bias equal to the step-size towards the best solution:

$$bias_2 = \sigma \frac{p_{best}}{\|p_{best}\|}$$

(4.21)

A threshold decision is included to select either $bias_1$ or $bias_2$. The threshold was chosen to be 10% of the measured distance of the CMA-ES mean value to the global best. If the step size of a CMA-ES instance exceeds that value, its mean is biased using $bias_2$. If it is below the threshold, $bias_1$ will shoot the run out of the local optimum. Yet another scenario must be considered. Runs that are exploring with a step-size larger than $\|p_{best}\|$ should not be biased, too, because they would overlap the target. Making it easier to keep apart all the different configurations, a Pseudo-Code is given in Algorithm 2.

The strength of $bias_1$ was tested on several functions with values between 0 and 1 as well as with a bias strength selected randomly, while the value of 0.5, as stated in Equation (4.20), was found to be the most stable choice. The same procedure counts for the threshold value. Nevertheless, there might exist more sophisticated biasing rules that will be investigated in future work.

### 4.4 Strategy Work-Flow and Parameter Setting

With the additional adaptation rule of $C$ and the bias of each CMA-ES run’s mean value, two SI-based operations foster global search characteristics of CMA-ES. Nevertheless, an algorithm with the ambition to be competitive in performance, necessitates further inquiries, concerning strategy parameters and work flow. So far, at least the following issues require sophisticated examination:

- What kind of information needs to be exchanged between CMA-ES runs?
- How often should SI operations influence the behaviour of CMA-ES?
- To what extend should these operations affect the search?

\footnote{The author is aware of the fact, that $\sigma$ only rescales the multivariate normal distribution and is not a direct measure for its extension. Still, it is a simple and intuitive scale-indicator of $C$.}
4.4 Strategy Work-Flow and Parameter Setting

4.4.1 Information Exchange between CMA-ES Instances

The two SI-based operations shall capitalize information obtained by the swarm. As PSO is a decentralized method, without an organizing global instance, communication between swarm members is essential. In PSO, different approaches regarding the inter-particle communication exist. The two most common ones are fully linked communication, with each swarm member communicating with every other individual, while another technique only allows exchange of information in a defined neighborhood. In PS-CMA-ES, all swarm members communicate with each other. With swarm size \( p \) this implies that \( p(p - 1) \) communication steps are needed. Though communication expenses seem to be quite big in such a setting, a moderate swarm size (a swarm size of 15 was mostly used during tests) relativizes the effort.

To both rotate the covariance matrix and bias the mean value, the global best position \( p_g \) needs to be known by every swarm member. Then, every instance can calculate its personal global best vector \( p_{best} = p_g - m \), rotate the covariance matrix and bias the mean value as described above. \( p_g \) is of course depending on the current best objective function value \( f_{best} \). Therefore, all current best function values of the swarm members need to be collected and ranked. Based on that ranking, instances can request \( p_g \) from the member which provides the global optimum.

Figure 7 illustrates the communication between swarm members. A swarm of size 4 was considered exemplarily. First, all particles exchange their personal optimum value \( f_{best} \) with each other (Figure 7(a)). Each individual ranks the collected values and swarm member number 4 was found to provide the best fitness value. Therefore it broadcasts the corresponding position, which is \( p_g \), to all its associates (Figure 7(b)). A more technical description of the communication process based on MPI routines can be found in section 7.1.1.

Note, that no other information than \( p_g \) and \( f_{best} \) is exchanged, which is a welcome
feature of PSO. To reduce communication expenses, \( p_g \) is only transmitted, when \( f_{\text{best}} \) changes.

Communication is also needed if one or more runs terminate. This can easily happen, if the CMA-ES instances have converged or met any stop criteria. Those runs need to be excluded from any further communication. This can be tricky, if a run that stops holds the current global best. It will not be able to communicate its position to the other swarm members in the next generation. The implementation of this issue is given in detail in section 7.1.2.

![Diagram](image.png)

_Figure 7: PS-CMA-ES Global Best Communication in a Swarm of Size 4_

### 4.4.2 Frequency of PSO Update

Another crucial design criterion is the frequency of PSO updates, i.e. how often the covariance matrix rotation and the mean bias should be applied during the algorithm’s runtime. For example, an update in every single generation would not be recommendable, as CMA-ES would have no time left to evolve and to approximate an objective function model. It would also not be eligible to perform PSO operations very rarely, as global information would hardly have any influence on the search evolution. It might even happen, that CMA-ES runs have already converged and terminated, before swarm information has had any impact on them. Therefore, a trade-off has to be found, such that CMA-ES has sufficient time to evolve without being distracted by SI-techniques, but also pays attention, that rotation and bias do change search characteristics.

### 4.4.3 Weighting the Covariances

Lastly, a third problem needs to be tackled. How strong should the influence of the rotated covariance matrix, defined by the factor \( c \) in Equation (4.1), be. Similar to the problem of frequency, the adjustment must ensure a convenient impact power of \( C_{PSO} \).

\( c \) can be interpreted as a weighting factor, defining up to what extend \( C_{CMA} \) is rotated towards \( C_{PSO} \). As discussed before, setting \( c = 0 \) is equal to a pure CMA-ES update. And \( c = 1 \) only accounts for a PSO based adaptation. Therefore, the limit values can be excluded in a hybrid method. A first approach could be to randomize \( c \).

In this thesis we use grid search to find a suitable parameter setting. Strategy parameters of the new approach are the swarm size, the frequency of PSO updates, the
4.4 Strategy Work-Flow and Parameter Setting

weighting factor $c$ and CMA-ES’s initial $\sigma$. A detailed description of the grid search is given in section 5.2.1.

4.4.4 Merging the Work Flows

We have explained how to calculate the rotated covariance matrix and how to reasonably bias the mean value of CMA-ES runs. We have described the communication between CMA-ES instances and extracted crucial strategy parameters. To complete the specification of PS-CMA-ES, its work flow is depicted.

The flow chart in Figure 8 illustrates the merging of the CMA-ES generation loop (see Figure 2) with the communication and PSO operations. Newly-created methods to integrate PSO into CMA-ES are highlighted in a light red. Next to the diagram entities, related equations and algorithms are displayed.

After initialization, candidate solutions are created, selected and recombined, yielding the mean value $m^{(g+1)}$. Now communication of $p_g$ takes place to keep the swarm members updated. Comparing the current global best to their own position, CMA-ES instances bias their mean value according to Algorithm 2. Note, that the bias changes the evolution path update in the succeeding generation, as the path is then calculated with reference to the biased mean value. After biasing, the covariance matrices $C_{CMA}^{(g+1)}$ and $C_{PSO}^{(g+1)}$ are calculated and combined, using Equation (4.1). This is followed by the step-size adaptation mechanism and the checking of stop criteria just as in the original CMA-ES flow chart. If a run stops, a revision of interconnections is needed to exclude swarm members from communication, before the next iteration can be started with creating the sample points of the next generation.
4.4 Strategy Work-Flow and Parameter Setting

Create Sample Points

Selection, Recombination and Paths Update

Global Best Communication

Bias Mean

Covariance Matrix Adaptation

Combine Covariances

Sigma Update and Error management

Stop Criteria?

Adapt Communicator

STOP CMA-ES RUN

Figure 8: PS-CMA-ES Generation Loop Chart and related Equations
5 Benchmark

5.1 Evaluation Criteria

5.1.1 Test Functions

The proposed PS-CMA-ES algorithm has been tested on 25 benchmark functions. They are provided by Suganthan et al. [29] during the CEC 2005 Special Session on Real-Parameter Optimization. The test suite aims at defining a standard benchmark for real-parameter optimization algorithms. Apart from the test functions, their main characteristics (uni/multimodal, separability, scalability, ...), and search space settings (box boundaries, initialization instruction, ...) are given. Furthermore, a systematic evaluation procedure (termination criteria, problem size, success criterion, etc.) is specified, in order to standardize the evaluation and to facilitate a comparison with other strategies.

Functions f1-f5 are unimodal and f6-f12 multi-modal test functions. f13-f14 are expanded functions and f15-f25 are hybrid test cases that result from the composition of several standard benchmark functions. To prevent exploitation of search space symmetry, all specified problems are shifted and many of them are rotated. The global optimum of each function is different from the common zero value.

5.1.2 Benchmark Settings

According to the evaluation criteria of the test suite, the method is benchmarked in \( n = 10, 30 \) and 50 dimensions. For each minimization problem, 25 runs are performed with a uniform random initialization within the search space. The maximum number of function evaluations (MAX_FES) is set depending on the problem dimension: \( \text{MAX}_\text{FES} = 10^4 n \). A run is terminated before reaching \( \text{MAX}_\text{FES} \), if the function error value is equal or below \( 10^{-8} \). As the global optimum of each objective function is known, the function error value is the absolute difference between the optimum and the achieved function value:

\[
   f_{err}(x) = |f(x) - \text{optimum}|
\]  

(5.1)

For each run, the function error value after \( 10^3, 10^4, 10^5 \) function evaluations (FES) and at termination is recorded.

<table>
<thead>
<tr>
<th>Problems</th>
<th>f1 - f25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runs/Problem</td>
<td>25</td>
</tr>
<tr>
<td>( n )</td>
<td>10, 30, 50</td>
</tr>
<tr>
<td>MAX_FES</td>
<td>( 10000 \cdot n )</td>
</tr>
<tr>
<td>Termination</td>
<td>If FES = MAX_FES or ( f_{err}(x) \leq 10^{-8} )</td>
</tr>
<tr>
<td>Initialization</td>
<td>Uniform random position</td>
</tr>
</tbody>
</table>

Table 1: Benchmark Settings according to the CEC 2005 Test Suite [29].
5.1 Evaluation Criteria

5.1.3 Success Performance

In [26] Auger and Hansen stated, that some algorithms may have a small probability of success but converge fast, while others may be slower, but with a larger probability of success. To evaluate success performance, two quantities are defined in [29]. The Success Rate, which is an estimator for the probability of success, and the Success Performance as an estimator for the number of function evaluations during successful runs:

\[
\text{Success Rate} = \frac{\text{\#successful runs}}{\text{\#runs}} \quad (5.2)
\]

\[
\text{Success Performance} = \frac{\text{mean(FES for successful runs)} \cdot \text{\#runs}}{\text{\#successful runs}} \quad (5.3)
\]

A run is successful, if the function error value reaches a given accuracy level (Table 2). Using these quantities, different optimization procedures can be easily compared.

<table>
<thead>
<tr>
<th>Function</th>
<th>Accuracy</th>
<th>Function</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$-450 + 1e-6$</td>
<td>14</td>
<td>$-300 + 1e-2$</td>
</tr>
<tr>
<td>2</td>
<td>$-450 + 1e-6$</td>
<td>15</td>
<td>$120 + 1e-2$</td>
</tr>
<tr>
<td>3</td>
<td>$-450 + 1e-6$</td>
<td>16</td>
<td>$120 + 1e-2$</td>
</tr>
<tr>
<td>4</td>
<td>$-450 + 1e-6$</td>
<td>17</td>
<td>$120 + 1e-1$</td>
</tr>
<tr>
<td>5</td>
<td>$-310 + 1e-6$</td>
<td>18</td>
<td>$10 + 1e-1$</td>
</tr>
<tr>
<td>6</td>
<td>$390 + 1e-2$</td>
<td>19</td>
<td>$10 + 1e-1$</td>
</tr>
<tr>
<td>7</td>
<td>$-180 + 1e-2$</td>
<td>20</td>
<td>$10 + 1e-1$</td>
</tr>
<tr>
<td>8</td>
<td>$-140 + 1e-2$</td>
<td>21</td>
<td>$360 + 1e-1$</td>
</tr>
<tr>
<td>9</td>
<td>$-330 + 1e-2$</td>
<td>22</td>
<td>$360 + 1e-1$</td>
</tr>
<tr>
<td>10</td>
<td>$-330 + 1e-2$</td>
<td>23</td>
<td>$360 + 1e-1$</td>
</tr>
<tr>
<td>11</td>
<td>$90 + 1e-2$</td>
<td>24</td>
<td>$260 + 1e-1$</td>
</tr>
<tr>
<td>12</td>
<td>$-460 + 1e-2$</td>
<td>25</td>
<td>$260 + 1e-1$</td>
</tr>
<tr>
<td>13</td>
<td>$-130 + 1e-2$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Fixed accuracy level according to the CEC 2005 Test Suite [29].

5.1.4 Algorithm Complexity

3 metrics are defined in [29] to measure the complexity of an optimization algorithm: \(T0\) reflects the computing time for standard operations. \(T1\) is given as the time to compute function \(f3\) for 200000 times in a certain dimension and \(T2\) is the mean value of 5 executions of the complete algorithm with 200000 FES again on function \(f3\).
5.2 Experimental Procedure and Results

5.2.1 Grid Search

In chapter 4, the swarm size $S$, the covariance weight $c$, the frequency of PSO updates and the initial step-size $\sigma$ have been introduced as crucial strategy parameters. An integral part of this thesis is the determination of good parameter settings. For this purpose we use extensive grid search.

The swarm size $S$ does not only influence the algorithmic behaviour, but is also hardware dependent. It is related to the overall population size and to the covering of the fitness landscape during search. The swarm size $S$ should be chosen low due to computational complexity but at the same time big enough to obtain useful global information.

For the weighting factor $c$ (see section 4.4.3) the limit values 0 and 1 can be discarded. Preliminary runs also showed, that setting $c = 0.1$ or $c = 0.9$ does not contribute to the algorithm in a favorable way.

If the frequency of PSO updates is set to 0, no communication between swarm members takes place. This setting resembles a Parallel CMA-ES and can be used as a reference to disclose, whether communication between CMA-ES instances is beneficial. A frequency value of 200, for example, indicates, that a PSO update is done at every 200th generation.

The strategy parameters for single CMA-ES instances are set according to [28]. Only the initial step-size $\sigma$ is adapted problem-dependently. We vary $\sigma$ between 20% and 50% of the constrained region of the minimization problem. For example, a test function is constrained to the boundary interval $[-5, 5]^n$. A $\sigma$ value of 0.3 would indicate an initial step-size of $0.3 \cdot (5 - (-5)) = 3$.

Note, that benchmarking PS-CMA-ES requires additional communication procedures. A global instance needs to count the overall function evaluations to log the number of evaluations after $10^3$, $10^4$ and $10^5$ FES and at termination to evaluate the success performance.

Table 3 summarizes all strategy parameters for the grid search. PS-CMA-ES is tested for every possible combination of parameters. If all possible parameter configurations are considered, 144 combinations need to be tested for each function. The benchmark, according to [29], requires 25 runs per problem. Hence, $144 \cdot 25 = 3600$ executions of PS-CMA-ES are needed for each function. Since the grid search is very extensive, it is restricted to $n = 10$ dimensions. For $n = 30$ and 50 dimensions, the parameter setting is used, that has been identified to give competitive performance on all 25 test functions in $n = 10$ dimensions.

Of course, the grid search could be further extended. More strategy parameter values could be tested or the search could be extended to 30 and 50 dimensions. Nevertheless, the values we used, are a first approach, yielding promising results, that are outlined in the next section.

---

5 The benchmark was done on a MacPro with 2 Dual-Core Processors (3 GHz). With a maximum swarm size of 15, the computational complexity was tolerable.

6 $4\sigma \cdot 3$ Swarm Size $\cdot 3c \cdot 4$ Frequency $= 144$ combinations.
5.2 Experimental Procedure and Results

Parameter | Value  
---|---
\(\sigma\) | 0.2, 0.3, 0.4, 0.5  
Swarm Size \(S\) | 6, 10, 15  
Weighting Factor \(c\) | 0.3, 0.5, 0.7  
Update Frequency | 0, 150, 200, 250  

Table 3: Strategy Parameter Grid Search. Configuration for 25 Test Functions in \(n = 10\) dimensions.

5.2.2 Results

An initial step size of \(\sigma = 0.2\), a swarm size of \(S = 15\), a weighting factor of \(c = 0.7\) and a PSO-inspired update every 200\(^{th}\) generation has been identified to perform well on most of the test functions. Though there might be better configurations for singular test cases, this configuration is chosen to be standard for analyses and subsequent tests in 30 and 50 dimensions. All following data are based on this specific parameter configuration.

For \(n = 10\) dimensions, 144 different parameter configurations have been tested and compared to PSGES, IPOP-CMA-ES and LR-CMA-ES. Results for these algorithms were taken from the related publications [25, 26, 46]. If the mean function error value after \(10^5\) FES is considered as a comparative value, like in [46], several statements can be made.

On the unimodal functions (f1–f5), no improvement compared to LR-CMA-ES and IPOP-CMA-ES is observed. Even more, configurations without communication between swarm members (Parallel-CMA-ES) converge faster, with better or equal performance. PSGES performs better on all unimodal test cases as well. Only on function f4, both the non-communicative and the particle swarm approach outperform LR-CMA-ES and behave like IPOP-CMA-ES.

On the standard multi-modal functions (f6–f12) results are ambivalent. While PS-CMA-ES is only superior to PSGES on functions f6–f8, it clearly outperforms the well-established CMA-ES algorithms and PSGES on f9–f12. It is found, that swarm communication is highly advantageous on f9. For f10–f12, Parallel-CMA-ES solved the problems with similar accuracy. On f9, f10 and f12 PS-CMA-ES reaches a success rate of 100%. On f11 the rate is still 40%. In comparison, LR-CMA-ES is only able to solve f12 with a probability of 48%. IPOP-CMA-ES is capable of solving f9–f12 with a success rate between 76% and 92%. For test case f11, IPOP-CMA-ES’s success rate is 24%. Comparing the success performance shows, that PS-CMA-ES also needs less objective function evaluations to find the global optimum on these functions.

On the hybrid test functions, PS-CMA-ES is a noteworthy competitor. It clearly outperforms PSGES and LR-CMA-ES. The latter only beats PS-CMA-ES on f20 and f21. IPOP-CMA-ES is inferior on f13, f15–f17 and f22. Furthermore, our method is the only algorithm that is able to find the optimum on f15 (success rate of 96%). Hence, it is the only algorithm that is able to solve a hybrid composition function among all competitors. Compared to Parallel-CMA-ES, the incorporation of global information through SI is favourable on hybrid problems.
Table 11 (Appendix B) shows the best objective function error values reached in dimension $n = 10$ for 25 benchmark functions, using the standard parameter configuration. Success evaluation for all tested problem dimensions is displayed in Table 4. After the grid search in 10 dimensions, the standard parameter configuration has been used to benchmark PS-CMA-ES in $n = 30$ and 50 dimensions\(^7\).

For $n = 30$, the new method outperforms its competitors regarding the mean error function value (Table 12) after $10^5$ FES on f4. It is superior to IPOP-CMA-ES by 2 orders of magnitude and superior to LR-CMA-ES by 4 orders of magnitude. On test functions f1–f3 and f5–f8 PS-CMA-ES is inferior to its competitors.

PS-CMA-ES is superior to LR-CMA-ES on the remaining functions except for f21 and f23. IPOP-CMA-ES reaches a lower mean error value on f16, f21 and f23. On f14 and f15, both, IPOP-CMA-ES and PS-CMA-ES achieve the same function error value. In summary, PS-CMA-ES is superior to LR-CMA-ES on 11 of 13 hybrid test functions and inferior to IPOP-CMA-ES on only 3 hybrid test cases in $n = 30$.

Similar performance is observed for $n = 50$ (Table 13). Except for the noisy f4 case, PS-CMA-ES performs bad on the unimodal functions as well as on the standard multimodal functions f6–f8. Again it achieves remarkable results on f9–f12. It beats LR-CMA-ES on the hybrid functions, except for f13 and f21. PS-CMA-ES is superior to IPOP-CMA-ES on f17–f24.

If we only look at the success evaluation, PS-CMA-ES performs rather bad for 30 and 50 dimensions. While IPOP-CMA-ES is still able to solve 11 problems in 30 dimensions and 7 for $n = 50$, PS-CMA-ES only solves 6 and 4, respectively\(^8\). Nevertheless, the remarkable results on functions f9 and f10 are not only reflected in the lower mean error value, but also in the success rate and success performance. In 30 and 50 dimensions, PS-CMA-ES was always able to solve these tasks with a probability of 100% and an expected number of FES between $2.49e + 04$ and $4.69e + 04$. Thus, it clearly outperforms IPOP-CMA-ES. The success rate of IPOP-CMA-ES drops to 12% and its success performance exceeds the values of PS-CMA-ES by a factor of more than 10 on f9 and f10.

Based on the mean function error value after $10^5$ FES, the different approaches can be compared, using a single value. The ranks are given in the following tables (Tables 5, 6 and 7) and a mean value (cumulated rank/#problems) was calculated for each approach [46].

\(^7\)Data for PSGES was only available for $n = 10$. Therefore PS-CMA-ES is only compared to LR-CMA-ES and IPOP-CMA-ES in the remaining test dimensions.

\(^8\)Please note, that none of the proposed algorithms was able to solve the test cases f13–f25 with defined tolerance.
Table 4: Performance measure for successfully optimized problems. Prob.: Problem Number; columns 2–8: number of function evaluations (minimal, 7th, median, 19th, maximal, mean and standard deviation) to reach given accuracy; columns 9–10: success rate and success performance.

The mean rank value is used to summarize the results. PS-CMA-ES is clearly inferior to other strategies on unimodal functions, except on f4. On multi-modal functions it is competitive and mostly ranked better. Especially in higher dimensions (n = 30, 50), the novel approach manifests its strengths, as it is ranked first on 14 of 20 multi-modal functions in n = 30 and on 11 of 20 in n = 50. Taking into account all 25 test functions, the performance (by means of ranking) of PS-CMA-ES is placed between IPOP-CMA-ES and LR-CMA-ES.

The computational complexity is displayed in Tables 8 – 10. We compare the complexity between a single CMA-ES run, Parallel CMA-ES and PS-CMA-ES. For Parallel CMA-ES and PS-CMA-ES a swarm of size 4 is used, because of the quad-core system architecture. It is observed, that \( \hat{T} \)2 of Parallel CMA-ES is nearly equal to the corresponding value of Standard CMA-ES. Compared to PS-CMA-ES, computational complexity also seems to be nearly equal up until n = 30. But for n = 50, computational expenses for PS-CMA-ES drastically increase.
5.2 Experimental Procedure and Results

<table>
<thead>
<tr>
<th>Function</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSGES</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>LR-CMA-ES</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>IPOP-CMA-ES</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>PS-CMA-ES</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSGES</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>3.4(85/25)</td>
</tr>
<tr>
<td>LR-CMA-ES</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>2.68(67/25)</td>
</tr>
<tr>
<td>IPOP-CMA-ES</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1.8(45/25)</td>
</tr>
<tr>
<td>PS-CMA-ES</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2.12(53/25)</td>
</tr>
</tbody>
</table>

Table 5: Ranking based on the mean function error value after $10^5$ FES for $n = 10$.

<table>
<thead>
<tr>
<th>Function</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR-CMA-ES</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>IPOP-CMA-ES</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>PS-CMA-ES</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR-CMA-ES</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2.32(58/25)</td>
</tr>
<tr>
<td>IPOP-CMA-ES</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1.84(46/25)</td>
</tr>
<tr>
<td>PS-CMA-ES</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1.76(44/25)</td>
</tr>
</tbody>
</table>

Table 6: Ranking based on the mean function error value after $10^5$ FES for $n = 30$.

<table>
<thead>
<tr>
<th>Function</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR-CMA-ES</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>IPOP-CMA-ES</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>PS-CMA-ES</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR-CMA-ES</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2.4(60/25)</td>
</tr>
<tr>
<td>IPOP-CMA-ES</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1.72(43/25)</td>
</tr>
<tr>
<td>PS-CMA-ES</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1.84(46/25)</td>
</tr>
</tbody>
</table>

Table 7: Ranking based on the mean function error value after $10^5$ FES for $n = 50$.

<table>
<thead>
<tr>
<th>System</th>
<th>Mac OS X 10.4.11</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>2x Dual-Core Intel Xeon 3.00GHz</td>
</tr>
<tr>
<td>RAM</td>
<td>1GB</td>
</tr>
<tr>
<td>Language</td>
<td>Fortran 90</td>
</tr>
<tr>
<td>Algorithm</td>
<td>CMA-ES</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0</td>
<td>3.02e-1</td>
<td>2.71e+0</td>
<td>2.53e+1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>9.53e-2</td>
<td>3.04e+0</td>
<td>2.51e+2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8: System configuration and measured CPU time for Standard CMA-ES

<table>
<thead>
<tr>
<th>System</th>
<th>Mac OS X 10.4.11</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>2x Dual-Core Intel Xeon 3.00GHz</td>
</tr>
<tr>
<td>RAM</td>
<td>1GB</td>
</tr>
<tr>
<td>Language</td>
<td>Fortran 90</td>
</tr>
<tr>
<td>Algorithm</td>
<td>Parallel CMA-ES</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0</td>
<td>3.02e-1</td>
<td>3.96e+0</td>
<td>3.84e+1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>9.53e-2</td>
<td>1.39e+1</td>
<td>1.22e+2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 9: System configuration and measured CPU time for Parallel CMA-ES
5.3 Discussion

It is noticeable that PS-CMA-ES is not competitive on uni-modal functions. Even more, with the standard parameter setting, it is not able to solve f3 in 10 dimensions. For \( n = 30 \), PS-CMA-ES only reaches the optimum on f1 and f2, while for \( n = 50 \), the only unimodal function solved was f1. There are several reasons for this astonishing behaviour. First, the parameter sweep for \( n = 10 \) indicates, that the standard setting is not optimal for unimodal functions. It is observed, that runs without communication and even smaller swarm sizes, perform better. This is because pure CMA-ES can well approximate the fitness landscape on unimodal functions and communication distracts its search. Second, one has to take into account, that there is only little time left for CMA-ES instances to converge within the maximum number of function evaluations, if the swarm size is large. The overall population size is swarm size \( \times \) CMA-ES population size. Consider a swarm size of 6, for example. Each CMA-ES instance can only do 1/6 of the function evaluations, that a singular CMA-ES run uses. When tackling high-dimensional tasks, the expected number of function evaluations grow. Therefore, it is plausible, that PS-CMA-ES reveals bad performance and low convergence speed, compared to standard CMA-ES. Generally, convergence speed is lower compared to other algorithms, due to the PSO update rules. If CMA-ES runs are biased by global information, the current approximation of the fitness landscape might not be appropriate for the new sample positions. Hence, CMA-ES needs to readjust its second-order model of the objective function from scratch.

On multi-modal functions, PS-CMA-ES is a serious competitor. The author believes, that its strengths open up on functions with a huge number of local optima (Rastrigin function, f9, f10) and on objective functions with no global structure (Schwefel problem, f12, hybrid composition functions). Furthermore, we detect, that PS-CMA-ES is very good on noisy functions. On both test cases with white gaussian noise in the fitness (f4 and f17), our method outperforms IPOP-CMA-ES and LR-CMA-ES in all test dimensions.

It was explained in Chapter 2, that CMA-ES is able to learn an underlying global structure of multi-modal functions, when using a sufficiently large population size. When there is no global fitness topology, as it is the case for the hybrid test functions, PS-CMA-ES might be the method of choice. Argumentation for the encouraging achievements of PS-CMA-ES on multi-modal problems, reflects a welcome feature of the method: exploration power.

The visualization of all candidate solutions of PS-CMA-ES during the search process on the shifted Rastrigin function (f9), clarifies the meaning of exploration power. The sample points for both, PS-CMA-ES and standard CMA-ES, are shown in Figure 9.
5.3 Discussion 5 Benchmark

(a) Standard CMA-ES

(b) PS-CMA-ES

Figure 9: Sampling Points during Evolution of PS-CMA-ES and Standard CMA-ES in $n = 2$. For PS-CMA-ES Swarm Size = 3 and PSO Frequency = 5. For CMA-ES Population Size = 3×Default Value.

(2-dimensional case). PS-CMA-ES was started with a swarm size of 3, each swarm member initialized at random, and a PSO update frequency of 5. To keep both approaches comparable, the population size for the standard CMA-ES run was set three times larger than the default value (default value × swarm size). So the overall population size remained the same for both strategies. By inspection, PS-CMA-ES covers a widespread region of the search space and is therefore more likely to find the optimum. Although CMA-ES finds the optimum for this simple test case, it is noticed, how its evolution is restricted to only a narrow scope of the fitness landscape. If the swarm size of PS-CMA-ES is increased, the covered range will even grow larger$^9$. Another observed aspect of this test is, that PS-CMA-ES reduced the number of samplings being out of bounds by a factor of more than 2. While the standard CMA-ES procedure needed 329 sample points to learn the boundaries, PS-CMA-ES sampled out of bounds only 135 times.

Conclusively, exploration power is favourable for most multi-modal tasks. Standard CMA-ES is restricted to a much more narrow scope of the search region and might get stuck in local optima. The PSO features of PS-CMA-ES help to overcome such minima and facilitate the CMA-ES instance to explore more expedient regions. The parameter sweep for 10 dimensions supports this hypothesis. On multi-modal functions, PS-CMA-ES performs clearly better than Parallel-CMA-ES.

Some final remarks about the parameter setting: while the weighting factor of $c = 0.7$ seems to be a stable choice, the standard frequency of 200 is discussible. It might be

$^9$The exploration power is, of course, coupled with the initialization of swarm members. If runs are initialized at very close positions to each other, exploration power will be less.
more reasonable to couple the frequency with the problem dimension instead of using a fixed frequency. In $n = 2$ dimensions, for example, CMA-ES has already converged in most of the cases before reaching generation 200. On the other side, CMA-ES needs a longer period to converge in 50 dimensions. A simple suggestion would be, to increase the frequency of updates for low-dimensional problems, and to lower it for high-dimensional optimization tasks.

Additionally, convergence speed is problem-dependent. On rather simple objective functions, CMA-ES converges faster than on difficult problems. The convergence speed directly influences the bias update rule (via $\sigma$). Therefore, a problem-dependent frequency of PSO updates might be more convenient. The author is confident, that a setting with consideration of problem characteristics can improve PS-CMA-ES’s outcome.

When talking about strategy parameters, the swarm size is not negligible. If the swarm consists of many individuals, more communication and update procedures are needed. This is reflected in the loss of computational speed. The amount of information, that needs to be communicated between swarm members is rather little. Therefore, communication might not heavily influence speed. But it is observed, that especially the rotation of the covariance matrix is a bottleneck. The rotation procedure (section 4.2.1) yields $n(n - 1) + 1$ matrix multiplications, that are needed to construct $R$ for each swarm member $^{10}$. Hence, computational speed drastically decreases in higher dimensions. As a recommendation, the swarm size should be chosen in compliance with hardware facilities. However, it is observed, that larger swarm sizes perform better on most multimodal functions. As an explanation, exploration power increases with growth of the swarm size and, hence, the likelihood to find the global optimum.

The intention of PS-CMA-ES is to supplement CMA-ES, which is considered as a local search method, with global information via PSO-like operations. The standard initial step-size confirms this concept. A small step-size stresses local search characteristics of CMA-ES. Thus, it is not surprising, that the chosen step-size (20% of the constrained region) performs very well on nearly all test cases. Generally, the standard parameter configuration not always yields the best results. However, the current setting is found to be robust on most of the problems and achieves remarkable results in all tested dimensions.

$^{10}$In fact, computational expenses exceed this value, since costs for constructing the Givens Rotation matrix and for keeping $p$ and $b$ updated were not considered (see Algorithm 1).
6 The Rate Balancing Problem in the Linear Multi-User MIMO Downlink

To complete this work, the PS-CMA-ES is applied to a real-world test case from the communications research area. The strategy is used to solve a rate balancing problem in the Multiple-Input-Multiple-Output Broadcast Channel (MIMO-BC) and is compared to Sequential Quadratic Programming (SQP) [49], Block Diagonalization (BD) [50], Gradient Minimum Mean Square Error (Grad-MMSE) and Gradient BD (Grad-BD) [51].

6.1 The MIMO-BC

Multiple-Input and Multiple-Output systems use multiple antennas at both the transmitter and receiver to improve communication performance. They received much attention as a method to achieve very high data rates over wireless links [17]. MIMO systems take advantage of multi-path signal propagation. That is, when different signals, due to scattering, reflexion, etc., arrive at the receiver at different times. Special signal processing is needed at the receiver to sort out the multiple signals to produce one signal, containing the originally transmitted data.

In a MIMO-BC system setup, a single base station with multiple transmit antennas is used to operate multiple users. Precoding at the base station is needed to maximize the received signal at a user. The following sections introduce and tackle a well-specified and self-contained rate balancing problem, proposed by Brehmer [15].

6.1.1 System Model

We consider a broadcast channel with one transmitter and \( K \) receivers (users). The base station has \( N \) transmit antennas and the receiver \( k \) has \( M_k \) receive antennas. The channel to user \( k \) is described by the channel matrix \( H_k \in \mathbb{C}^{M_k \times N} \). After precoding the data signal \( s_k \in \mathbb{C}^{M_k} \) (with precoding matrix \( P_k \in \mathbb{C}^{N \times M_k} \)), the transmit signal for user \( k \) reads

\[
x_k = P_k s_k.
\] (6.1)

The overall transmitted signal is given as

\[
x = \sum_{k=1}^{K} x_k = \sum_{k=1}^{K} P_k s_k.
\] (6.2)

The signal is power-constrained. If statistically independent, unit-power and zero-mean data symbols are considered, the constraint can be expressed as a constraint on the precoding matrices:

\[
\sum_{k=1}^{K} \text{tr}(P_k^H P_k) \leq P_{\text{tot}}.
\] (6.3)
The received signal at user $k$ can be calculated as

$$y_k = H_k x + \eta_k = \sum_{l=1}^{K} H_k P_l s_l + \eta_k,$$  \hspace{1cm} (6.4)

where the noise at the receiver, $\eta_k$, is drawn from a circularly symmetric complex Gaussian distribution with zero-mean and covariance matrix $\mathbf{R}_{\eta_k} = \sigma^2 \mathbf{1}$.

### 6.1.2 Achievable Rate

The channel rate $r_k$, which depends on the choice of the precoding matrices, can be calculated as

$$r_k(P_1, \ldots, P_K) = \log_2 \frac{\det \left( \sigma^2 \mathbf{1} + \sum_{l=1}^{K} H_k P_l \mathbf{P}_l^H \mathbf{H}_k^H \right)}{\det \left( \sigma^2 \mathbf{1} + \sum_{l=1}^{K} H_k P_l \mathbf{P}_l^H \mathbf{H}_k^H \right)}.$$ \hspace{1cm} (6.5)

A mapping is used to derive a real-valued representation of the information rate [52]. The map $\hat{\mathbf{A}}: \mathbb{C}^{m \times n} \rightarrow \mathbb{R}^{2m \times 2n}$, $\mathbf{A} \mapsto \hat{\mathbf{A}}$ is given as

$$\hat{\mathbf{A}} = \begin{bmatrix} \text{Re}(\mathbf{A}) & -\text{Im}(\mathbf{A}) \\ \text{Im}(\mathbf{A}) & \text{Re}(\mathbf{A}) \end{bmatrix}. \hspace{1cm} (6.6)$$

The precoding matrices are vectorized, representing the optimization variables as

$$\mathbf{x} = (\text{vec}(\text{Re}(\mathbf{P}_1)), \text{vec}(\text{Im}(\mathbf{P}_1)), \ldots, \text{vec}(\text{Re}(\mathbf{P}_K)), \text{vec}(\text{Im}(\mathbf{P}_K))), \hspace{1cm} (6.7)$$

with $\mathbf{x} \in \mathbb{R}^L$ and $L = 2N \sum_{k=1}^{K} M_k$.

With a map $\pi : (\mathbf{x}, k) \mapsto \hat{\mathbf{P}}_k$, the desired map $r_k : \mathbb{R}^L \rightarrow \mathbb{R}$ is given as

$$r_k(\mathbf{x}) = \log_2 \frac{\det \left( \sigma^2 \mathbf{1} + \sum_{l=1}^{K} \hat{\mathbf{H}}_k \pi(\mathbf{x}, l) \pi(\mathbf{x}, l)^T \hat{\mathbf{H}}_k^T \right)}{\det \left( \sigma^2 \mathbf{1} + \sum_{l=1}^{K} \hat{\mathbf{H}}_k \pi(\mathbf{x}, l) \pi(\mathbf{x}, l)^T \hat{\mathbf{H}}_k^T \right)}.$$ \hspace{1cm} (6.8)

The sum-power constraint turns into

$$||\mathbf{x}||^2 \leq P_{\text{tot}}.$$ \hspace{1cm} (6.9)

### 6.2 Rate Balancing

We apply PS-CMA-ES to the rate balancing problem. Multiple users may ask for a specific data rate. Since the overall rate is power-constrained, it might not be possible to allow users the rate they require. In that case, user rates need to be balanced, such
that no user is neglected and every user is allocated a maximal possible rate at the same time. Mathematically, the rate balancing problem for a given target rate $R_k^{\text{target}}$ for user $k$ reads

$$\max_{x \in \mathbb{R}^L, \gamma} \gamma \quad \text{s.t.} \quad r_k(x) = \gamma R_k^{\text{target}}, \quad k = 1, \ldots, K, \quad ||x||^2 \leq P_{\text{tot}}. \quad (6.10)$$

By solving the rate balancing problem for different targets, we can trace an achievable rate region, which is a true subset of the so-called capacity region. The capacity region of a MIMO-BC provides an upper bound on the rate vectors $r = (r_1, \ldots, r_K)$, achievable by linear precoding.

We considered a $K = 2$ user system with the following parameters:

\begin{align*}
K &= 2, \\
N &= 4, \quad M_1 = 2, \quad M_2 = 2, \\
P_{\text{tot}} &= 10, \quad \sigma^2 = 1, \\
H_1 &= \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 4 & 3 \end{bmatrix} + j \begin{bmatrix} 1 & 4 & 3 & 2 \\ 1 & 4 & 2 & 3 \end{bmatrix}, \\
H_2 &= \begin{bmatrix} 1 & 3 & 2 & 4 \\ 1 & 3 & 4 & 2 \end{bmatrix} + j \begin{bmatrix} 2 & 1 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{bmatrix}, \quad \text{with imaginary unit } j.
\end{align*}

### 6.3 Procedure

#### 6.3.1 Problem Formulation

A convenient formulation of the optimization problem is needed in order to handle the sum-power constraint. We define a hyper-sphere with maximal radius $r_{\max} = \sqrt{P_{\text{tot}}}$. All possible solution vectors $x$ reside within the sphere. By that it is guaranteed, that $||x||^2 \leq P_{\text{tot}}$.

There are two possibilities, how to map $x$ to such a hyper-sphere. One could transform $x$ into a polar coordinate system. But because of the transformation, the hyper-sphere is not sampled uniformly. Therefore, we follow [53]: a normally sampled vector $x$ can be mapped, such that the samples are uniformly distributed on a hyper-sphere. However, the radius of the sphere does not necessarily have to equal $\sqrt{P_{\text{tot}}}$. We introduce another optimization variable, $p$, which is used to optimize the system power. In the mapped hyper-sphere space, $p$ scales the radius, $r$, of the sampled hyper-sphere:

$$r = p \cdot r_{\max}, \quad \text{with} \quad p \in [0, 1]. \quad (6.11)$$

The mapping of $x$ is given as

$$x_{\text{mapped}} = r \cdot \frac{x}{||x||}. \quad (6.12)$$

$x_{\text{mapped}}$ is used as the input vector for Equation (6.8). The mapping allows to do unconstraint optimization ($x \in [-\infty, \infty]$). This is highly favourable, because CMA-ES does not need to learn any boundaries.
However, Equation (6.10) also contains the equality constraint $r_k(x) = \gamma R_k^{\text{target}}$. Hence, $x_{\text{mapped}}$ is only a feasible solution of the problem, if

$$\frac{r_k(x_{\text{mapped}})}{R_k^{\text{target}}} - \gamma = 0. \quad (6.13)$$

The rate balancing problem requires to find a maximal value for $\gamma \in [0, 1]$, such that Equation (6.13) holds. We do a line search on $\gamma$, starting from 1 in order to find that value for different targets $R_k^{\text{target}}$. Each target configuration defines a ray, emerging from the origin, and Equation (6.13) ensures, that the solution is always on that ray. The system parameters yield a problem dimension of $L = 2 \cdot 4 (2 + 2) = 32$. With the additional optimization variable $p$, dimensionality increases: $32 + 1 = 33$.

### 6.3.2 Parameter Setting

Both, Standard CMA-ES and PS-CMA-ES are applied. A run is stopped, if the fitness value is lower than $1e-7$ or if the number of FES exceeds $5e+5$. Preliminary tests showed, that the population size needs to be rather large, so it is set to $\lambda = 3L = 96$. Targets are chosen to be on a circular arc with radius 16 and an angular difference of $3^\circ$ (see Figure ??). Starting from 1, $\gamma$ is decreased by 0.02, if 3 runs in a row are not able to find a solution vector for Equation (6.13). Swarm parameters were chosen as: Swarm Size $S = 8$, Update Frequency = 200, $c = 0.7$ and $\sigma = 0.2$.

As a comparison, the problem is also solved with SQP, BD, Grad-BD and Grad-MMSE. Matlab’s `fmincon` function with standard medium-scale parameters is used as an implementation of SQP [54]. For SQP, the maximal number of FES is reduced to $1e+5$, since preliminary tests showed, that the maximal number of FES is never reached. Apart from that, the same setup as for PS-CMA-ES and Standard CMA-ES is used.

Data for the other optimization methods, BD, Grad-BD and Grad-MMSE is provided by Brehmer. 7 data points are available for those algorithms. Settings are unknown to the author.

### 6.4 Results and Discussion

Figure 10 shows the achieved rate regions for Standard CMA-ES, PS-CMA-ES and SQP. The capacity region (upper bound) is also displayed. It is noticeable, that the upper bound for the capacity region is not reached by any algorithm. It has to be noted, that the capacity bound is generally not reached, but is a theoretical upper bound.

However, the limit values are reached by Standard CMA-ES and PS-CMA-ES, if one of the target rates is 0. SQP reveals the worst performance for the rate limits.

Looking at other target configurations, we observe, that the hull of the achieved rate region of CMA-ES, PS-CMA-ES and SQP is not smooth. There are two reasons for that: first, the hull curve is spiky because of target and $\gamma$ resolution. Second, the spikes indicate, that an optimal solution is not found. We see, that all three optimization algorithms reveal nearly equal performance for most of the targets. Looking at the
data, it is also found, that the transmitter power for all targets equals $P_{\text{tot}}$.
A simple reference for the achieved rate region, can be found by solving the sum rate maximization problem:

$$\max_{\mathbf{x} \in \mathbb{R}^L} \sum_{k=1}^{K} r_k(\mathbf{x}) \quad s.t. \quad ||\mathbf{x}||^2 \leq P_{\text{tot}}$$  \hspace{1cm} (6.14)

The solution of the sum rate problem is $r_1 = 6.47$, $r_2 = 10.54$ (personal communication with Brehmer). A first reference for the achievable rate region is the convex hull given by the rate limits and the sum rate solution. This only counts, if we assume the sum rate solution to be a global optimum. It can be seen, that SQP, Standard CMA-ES and PS-CMA-ES come very close to that region and PS-CMA-ES nearly reaches the optimal solution for the sum rate problem.

Compared to BD, Grad-BD and Grad-MMSE (see Figure 11), the evolutionary algorithms and SQP perform very well. Only Grad-MMSE seems to be competitive and reveals nearly equal performance.

SQP is considered to be less complex and more efficient than the evolutionary algorithms. Therefore, it might be the method of choice. However, it has problems in finding the rate limits. Improvements of Standard CMA-ES and PS-CMA-ES could be made by tuning the strategy parameters, e.g. increase the population size.
Figure 10: Achieved Rate Region for PS-CMA-ES, Standard CMA-ES and SQP.
Figure 11: Achieved Rate Region for BD, Grad-BD and Grad-MMSE.
7 Code Design and Implementational Details

7.1 Distributed PS-CMA-ES using MPI

Nowadays, parallel code is frequently used for computationally expensive calculations. The Message Passing Interface (MPI) standardizes communication between multiple processes. It has been designed by a working group with participators from over 40 organizations and aimed at establishing a portable, efficient, and flexible standard for message passing [55]. The protocol makes it possible to perform parallel calculations on distributed, heterogeneous and loosely-coupled computer systems [56]. MPI is available for C and Fortran programs.

In this work, a Parallel CMA-ES code is developed based on MPI. Multiple CMA-ES instances need to communicate in order to integrate SI-techniques. MPI is designed to tackle such tasks on an implementational level. Each CMA-ES instance represents a computational process with separate address spaces. Via MPI, these processes can be distributed to multiple processors (for example to different PCs in a Local Area Network (LAN), see Figure 12). If such a topology is not available, MPI allows to execute parallel code on single processors.

Unfortunately, the developed algorithm is computationally demanding. Especially when multiple CMA-ES units have to share hardware facilities, computation speed decreases. Therefore, an elaborate communication strategy is essential to reduce complexity.

Figure 12: Multiple CMA-ES instances distributed in a Local Area Network (LAN).

7.1.1 GLOBAL_BEST-Communication

In MPI, each process is assigned a unique number: the process rank. Multiple processes can be distinguished by their rank. In section 4.4.1 it has been outlined, that each CMA-ES instance has to inform the other swarm members on its current best candidate
solution. Figure 13 illustrates our procedure. A 2-dimensional array, called F\_BEST, is introduced. It holds the current best fitness value, as well as the process rank (illustrated by light red ellipses in Figure 13(a)). The MPI collective communication routine \texttt{MPI\_ALLREDUCE} is able to find the global best function value within all F\_BEST arrays. Since the array also contains the process rank, the corresponding CMA-ES instance is known. In a second step (Figure 13(b)), this process broadcasts the position of its current optimal solution to the other swarm members.

To reduce communication expenses, the broadcast is only performed, when the current optimum has changed.

![Diagram](a) MPI Allreduce ![Diagram](b) MPI Broadcast

Figure 13: MPI Communication of the Global Best Position \( p_g \)

### 7.1.2 Excluding processes from communication

If a CMA-ES instance has converged and stopped its search, it needs to be excluded from communication. There are two possible scenarios, how a safe program termination can be ensured, and a communication deadlock can be avoided. One is, that the process waits until all other running processes have reached the same state, such that all processes can be finalized synchronously. Such an approach heavily influences performance, because resources keep on being allocated, until the last process has converged.

Therefore, another, more dynamic approach is favoured: whenever a process stops, communication is adapted, such that this specific process is excluded. To describe our method, definitions of MPI groups and communicators have to be given [57]:

**Definition 7.1** A \textit{group} is an ordered set of processes. Each process in a group is associated with a unique integer rank. Rank values start at zero and go to \( N-1 \), where \( N \) is the number of processes in the group. A group is always associated with a communicator object.

**Definition 7.2** A \textit{communicator} encompasses a group of processes that may communicate with each other. All MPI messages must specify a communicator. For example, the communicator that comprises all tasks is \texttt{MPI\_COMM\_WORLD}. 
In MPI, communicators can not be adapted directly. Therefore, a little workaround is needed to adapt communications. At the end of each CMA-ES generation, it is checked, whether one or more processes have met a stopping criterion and are about to terminate. If this is the case, the following rules are applied:

1. Ranks of terminating processes are collected.

2. *MPI_COMM_GROUP()* and *MPI_GROUP_EXCL()* are used to build a new communication group, that excludes the specified ranks.

3. A new rank for each process in the group (*MPI_GROUP_RANK()*)) is assigned.

4. A group communicator using *MPI_COMM_CREATE()* is created.

5. Calculations using the newly created communicator are continued. All processes, that are not in the scope of the communicator, are terminated.

Figure 14: Excluding processes from communication

Figure 14 illustrates an example for the MPI group and communicator management. At the very top of the figure, all processes are within the same communicator environment (*MPI_COMM_WORLD*). Colored in a light red are processes that will stop. Following the chart, two groups are formed. One group contains running processes (Group 1), the other contains stopping processes (Group 2). Based on Group 1, a new communicator can be created. Note, that the processes are assigned a new rank starting from 0 again.

\footnote{Group 2 is not a valid communication group. Instead it contains processes with rank value *MPI_UNDEFINED*}
7.2 Tutorial: Fortran (PS-)CMA-ES Library

7.2.1 Code Design

A major part of this thesis is the development of a Fortran CMA-ES Library. One reason to choose Fortran as an implementational language, is its outstanding performance on complex numerical operations. Together with the Linear Algebra Package (LAPACK), Fortran is a well-suited language for matrix operations. Due to the user-friendly MPI, Fortran code can be easily distributed and, hence, is predestinated for our task to create parallel, communicating CMA-ES instances. Another reason to use Fortran was, that so far no CMA-ES implementation in Fortran exists. The author hopes, that this work is a useful contribution to the community. Especially when we consider, that Fortran is an extensively used engineering language and standard in high performance computing.

All code is developed in accordance to the Fortran 90/95 standard [58]. A fundamental goal in the code design is to keep the code structured and easy-to-read. Clearly defined interfaces are needed to facilitate the integration of new modules, such as SI-operations. Furthermore, potential users should not have to bother about code structure, compilation, etc.

The CMA-ES implementation follows the Matlab version 2.54\textsuperscript{12}. In the following, some features of the code are described and an example of how to use the software is given.

7.2.2 Getting Started

Requirements. The current implementation of the CMA-ES Library has been tested on Windows XP SP2 and Mac OS X 10.4. Portability to other platforms is not guaranteed. In order to compile the CMA-ES Library, a Fortran compiler is necessary and the user should generally be familiar with the compilation of Fortran Code. Furthermore, an installed and running distribution of the Linear Algebra and the Basic Linear Algebra Subprogram Package (LAPACK/BLAS) [59] is essential.

The library is able to save output data in a Matlab-readable file format (.mat). To use this feature, certain Matlab libraries are needed, hence, Matlab must be installed on the system. If not available, no data will be lost, but instead be saved to .txt files. In order to use the PSO features of the library, a MPI distribution needs to be installed.

Configuration. Before compilation, some system-specific configurations have to be set. In the make.inc file, these settings can be adapted. This is useful, because no major changes have to be done to the makefile itself, when system configurations change and portability is improved.

A basic example. We examplarily set up a little program, that calls PS-CMA-ES to optimize the shifted Rastrigin function (f9 of the CEC benchmark). The program is called pscma_example and given in Appendix D.

First of all, we include the modules cmaes_param_mod and cmaesOpts_mod (line 3/4).

\textsuperscript{12}available at http://www.bionik.tu-berlin.de/user/niko/cmaes_inmatlab.html
They allow us to specify all options of PS-CMA-ES. We also include the CEC2005 module (l. 5), in which the test function `F09_shifted_rastrigin` is defined.

Since we want to use swarm operators, the necessary MPI header file is included in line 8–10. The interface `cmaesStart` (l.13–23) ensures a parameter check, when PS-CMA-ES is started. Next, variables are declared (l.27–30): `ierr`, used as a return value for MPI routines, the problem dimension `d`, the initial start position `xstart` and the initial step–size `insigma`.

In lines 35–41, MPI and the obligatory strategy parameters `xstart` and `insigma` are initialized. In the following (l.44–57), some options are set. A list of available options is given in Appendix E.

In line 60, the PS-CMA-ES is finally started with calling the `cmaesStart` method. The name of the objective function `F09_shifted_rastrigin` is the first parameter, followed by the starting position and the initial step–size. The options are already saved in the `cmaesOpts.mod` module and do not have to be passed to the function.

To compile the program together with the CMA-ES Library, the `make.inc` file needs to be adapted (see Appendix C). The comments should sufficiently explain the settings. Since we do not want to modify the makefile directly, we add the sourcefile (`pscma_example.f90`) via the variable `ADDSOURCES` (line 48). Typing `make all` in a console should start compilation and a file called `a.out` should be listed in the current directory afterwards.

The program can now be executed. To start multiple CMA-ES instances, the command `mpirun` is needed\(^\text{13}\): `mpirun -np 10 a.out`, for example, calls the program with a swarm size of 10.

**Adding objective functions.** Adding user-specific objective functions is rather simple. The easiest way is, to define the objective function via a module, as it has been done in the example. To compile the functions along with PS-CMA-ES, the `ADDSOURCES` and `ADDPATH` variables in the `make.inc` file can be used. When writing objective functions, the following header should be used:

```
function name(res,x,N,M,LBounds,UBounds)
  res Real Return value (fitness value), M-dim vector
  x Real Input vector, N×M array
  N Integer Problem Dimension
  M Integer Number of vectors to process
  LBounds Real Lower bound, optional, N-dim vector
  UBounds Real Upper bound, optional, N-dim vector.
```

However, the above function header might not be convenient for all optimization tasks. If the header needs to be changed (not recommended), the file `funcWrap.f90` can be adapted to personal needs. This function wrapper is the interface between PS-CMA-ES and any objective function. Only experienced users should change the wrapper.

\(^\text{13}\)Depending on the MPI distribution, the program call might look different.


8 Summary and Outlook

In this thesis a novel optimization methodology is introduced. Our method combines two well-known optimization procedures: the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) and Particle Swarm Optimization (PSO). CMA-ES is a highly elaborate optimization strategy, applied to non-linear, non-convex and continuous domain problems. It is considered to currently be the most powerful evolution strategy and is usually used, when gradient-based methods fail. CMA-ES particularly exhibits its strength as a local search method. PSO on the other hand, has been shown to perform well as a global search strategy. It obtained great acceptance because of its simplicity and reliable performance. In our approach, CMA-ES is augmented with PSO concepts in order to integrate global information to CMA-ES. Parallelization of CMA-ES instances enables us to make use of the remarkable local search characteristics of CMA-ES, while communication between different instances is used to exploit global information. We introduce two operators to adapt the evolution of CMA-ES: first, the covariance matrix is rotated, such that its principal axis points to the global best fitness position. Second, CMA-ES instances are biased towards that direction, based on their current position and state of evolution. Our hybrid approach, called Particle Swarm CMA-ES (PS-CMA-ES), is benchmarked on 25 test functions and compared to IPOP-CMA-ES, LR-CMA-ES and PSGES, in accordance with the CEC 2005 evaluation criteria [29]. Results show, that PS-CMA-ES, can leverage the different search characteristics of CMA-ES and PSO. It is found, that PS-CMA-ES performs especially well on multimodal, hybrid test functions with no global structure. Compared to other optimization strategies, PS-CMA-ES outperforms LR-CMA-ES and PSGES, although its convergence speed is slower in most cases. In 30 dimensions, PS-CMA-ES is ranked first and is only inferior to IPOP-CMA-ES in 10 and 50 dimensions. This is mainly because performance on unimodal functions is bad. On most of the multimodal functions, PS-CMA-ES is superior to its competitors. Exploration power, due to spatial distribution of CMA-ES instances in the search space, is found to be a welcome feature and significantly improves performance on hybrid test functions. A major emphasis of this work is placed on the retrieval of good strategy parameters. We discover, that a large swarm size is favourable, but increases complexity. The frequency of PSO updates and a weighting factor to balance the influence of the CMA-ES based covariance matrix and the PSO based covariance matrix, must be chosen reasonable.

Due to parallelization, communication and the rotation operator, PS-CMA-ES is a computationally complex, but promising strategy. With respect to the author’s background, PS-CMA-ES is also applied to a rate balancing optimization problem in the MIMO BC. It is shown, that PS-CMA-ES and Standard CMA-ES exhibit nearly equal performance and the upper bound of the capacity region is only found, if the target rate of one user equals 0. No significant improvement is found in comparison to SQP. However, BD, Grad-BD and Grad-MMSE are clearly outperformed. The test case also demonstrates, that the algorithm is ready-to-use and
can be applied to a wide range of problems.
All code has been developed in Fortran with stress on modularity, adaptability and portability. The implemented library can be used both as Standard CMA-ES and PS-CMA-ES, realized through the MPI.
There is room for improvement in future work. More research is needed to find an optimal strategy parameter configuration for unimodal functions. Plans to set parameters according to problem specifications (problem dimension, uni/multimodal, etc.), can surely improve the performance. A review of the bias update rules might be advantageous. It is observed, that the bias has the strongest influence on performance. However, the current setting yields promising results.
Parallel CMA-ES instances open-up the way to another approach: meta-evolution. Apart from SI-methods, evolutionary concepts could be applied on CMA-ES instances. For example, a selection operator could select good CMA-ES runs and erase bad runs. We consider each CMA-ES instance as a self-contained evolutionary system and refer to meta-evolution as a concept, that applies evolutionary methodologies to a population of such systems.
Because of the modular code design, such approaches can be easily integrated. We hope, that meta-evolution will improve performance on unimodal problems and make PS-CMA-ES converge faster. A first idea is, to erase unsuccessful runs and to re-initialize them somewhere else. On unimodal functions, the principal axis of CMA-ES runs usually points towards the optimum after a while. Runs can be ranked based on their best fitness value. For example, in a swarm of size 3, we erase the 3rd-ranked CMA-ES instance and re-initialize it at the intersection of the axis of the first and second-ranked runs.
Another idea is, to create a meta-CMA-ES instance, i.e. a CMA-ES run, whose initial candidate solutions are drawn from the population of best candidate solutions from other instances.
Tests have to show, whether meta-evolution is a useful concept. Current ideas are only rudimentary and will be developed in future work.
A Eigendecomposition of Covariance Matrix \( C \)

The Covariance Matrix \( C \in \mathbb{R}^{n \times n} \) is determined by its orthonormal basis of eigenvectors \( B = [b_1, \ldots, b_n] \) and corresponding eigenvalues \( d_1^2, \ldots, d_n^2 > 0 \). If \( D^2 \) is a diagonal matrix, containing the eigenvalues as diagonal elements, \( C \) can be decomposed as

\[
C = BD^2B^T. \tag{A.1}
\]

From (A.1) follows that

\[
C^{\frac{1}{2}} = BDB^T \tag{A.2}
\]

and

\[
C^{-\frac{1}{2}} = BD^{-1}B^T
= B \text{ diag} \left( \frac{1}{d_1}, \ldots, \frac{1}{d_n} \right) B^T. \tag{A.3}
\]

The inverse of \( C \) is calculated by

\[
C^{-1} = (BD^2B^T)^{-1}
= BD^{-2}B^T
= B \text{ diag} \left( \frac{1}{d_1^2}, \ldots, \frac{1}{d_n^2} \right) B^T. \tag{A.4}
\]
### Table 11: Best objective function error values reached after 10^5, 10^4 and 10^5 FES in dimension \( n = 10 \). Given are minimum, 7th, median, 19th and maximum value as well as mean value and standard deviation from 25 runs. PS-CMA-ES parameter configuration: Swarm Size = 15, \( \sigma = 0.2 \), Frequency = 200 and \( c = 0.7 \).

<table>
<thead>
<tr>
<th>FES</th>
<th>Prob.</th>
<th>f1</th>
<th>f2</th>
<th>f3</th>
<th>f4</th>
<th>f5</th>
<th>f6</th>
<th>f7</th>
<th>f8</th>
<th>f9</th>
<th>f10</th>
<th>f11</th>
<th>f12</th>
<th>f13</th>
</tr>
</thead>
<tbody>
<tr>
<td>1e3</td>
<td>min</td>
<td>1.85e+01</td>
<td>4.75e+01</td>
<td>2.21e+07</td>
<td>6.85e+03</td>
<td>4.46e+03</td>
<td>6.98e+07</td>
<td>1.20e+03</td>
<td>2.06e+01</td>
<td>5.10e+00</td>
<td>1.09e+01</td>
<td>1.06e+01</td>
<td>1.56e+04</td>
<td>6.77e+00</td>
</tr>
<tr>
<td>1e4</td>
<td>7th</td>
<td>3.65e+03</td>
<td>8.41e+03</td>
<td>4.34e+07</td>
<td>9.58e+03</td>
<td>6.35e+03</td>
<td>2.75e+08</td>
<td>1.21e+03</td>
<td>2.07e+01</td>
<td>7.18e+00</td>
<td>2.03e+01</td>
<td>1.16e+01</td>
<td>4.57e+04</td>
<td>9.01e+00</td>
</tr>
<tr>
<td></td>
<td>med</td>
<td>4.50e+03</td>
<td>1.15e+04</td>
<td>6.18e+07</td>
<td>1.18e+04</td>
<td>7.59e+03</td>
<td>3.64e+08</td>
<td>1.22e+03</td>
<td>2.08e+01</td>
<td>1.16e+01</td>
<td>2.37e+01</td>
<td>1.20e+01</td>
<td>5.76e+04</td>
<td>1.08e+01</td>
</tr>
<tr>
<td></td>
<td>19th</td>
<td>5.34e+03</td>
<td>1.27e+04</td>
<td>7.14e+07</td>
<td>1.60e+04</td>
<td>8.69e+03</td>
<td>6.47e+08</td>
<td>1.22e+03</td>
<td>2.08e+01</td>
<td>1.31e+01</td>
<td>2.79e+01</td>
<td>1.23e+01</td>
<td>6.87e+04</td>
<td>1.39e+01</td>
</tr>
<tr>
<td></td>
<td>max</td>
<td>7.73e+03</td>
<td>1.75e+04</td>
<td>1.23e+08</td>
<td>2.17e+04</td>
<td>1.03e+04</td>
<td>4.66e+09</td>
<td>1.30e+03</td>
<td>2.09e+01</td>
<td>1.97e+01</td>
<td>3.54e+01</td>
<td>1.34e+01</td>
<td>9.38e+04</td>
<td>1.84e+01</td>
</tr>
<tr>
<td></td>
<td>mean</td>
<td>4.86e+03</td>
<td>1.16e+04</td>
<td>6.44e+07</td>
<td>2.20e+04</td>
<td>7.35e+03</td>
<td>4.82e+08</td>
<td>1.23e+03</td>
<td>2.05e+01</td>
<td>1.16e+01</td>
<td>2.38e+01</td>
<td>1.20e+01</td>
<td>5.82e+04</td>
<td>1.12e+01</td>
</tr>
<tr>
<td></td>
<td>std</td>
<td>1.58e+03</td>
<td>3.11e+04</td>
<td>2.47e+07</td>
<td>1.42e+04</td>
<td>1.52e+03</td>
<td>7.77e+08</td>
<td>1.70e+03</td>
<td>8.68e+02</td>
<td>3.98e+00</td>
<td>5.78e+00</td>
<td>7.74e+01</td>
<td>1.94e+04</td>
<td>2.97e+00</td>
</tr>
<tr>
<td>1e5</td>
<td>min</td>
<td>6.17e+02</td>
<td>2.04e+01</td>
<td>5.66e+03</td>
<td>6.65e+01</td>
<td>2.14e+02</td>
<td>1.19e+02</td>
<td>1.96e+00</td>
<td>2.02e+01</td>
<td>2.73e+04</td>
<td>5.09e+04</td>
<td>2.00e+00</td>
<td>5.11e+01</td>
<td>1.66e+01</td>
</tr>
<tr>
<td>1e4</td>
<td>7th</td>
<td>1.94e+01</td>
<td>7.51e+01</td>
<td>1.16e+06</td>
<td>1.20e+02</td>
<td>4.33e+02</td>
<td>2.86e+02</td>
<td>1.26e+00</td>
<td>2.05e+01</td>
<td>4.92e+04</td>
<td>1.84e+03</td>
<td>3.95e+00</td>
<td>1.18e+02</td>
<td>2.32e+00</td>
</tr>
<tr>
<td></td>
<td>med</td>
<td>2.77e+01</td>
<td>1.10e+02</td>
<td>1.76e+06</td>
<td>1.68e+02</td>
<td>5.19e+02</td>
<td>7.93e+02</td>
<td>1.36e+00</td>
<td>2.06e+01</td>
<td>6.77e+04</td>
<td>2.27e+03</td>
<td>4.80e+00</td>
<td>1.84e+02</td>
<td>2.66e+00</td>
</tr>
<tr>
<td></td>
<td>19th</td>
<td>3.93e+01</td>
<td>2.91e+02</td>
<td>2.91e+06</td>
<td>5.48e+02</td>
<td>1.25e+03</td>
<td>1.50e+00</td>
<td>2.06e+01</td>
<td>8.66e+04</td>
<td>2.83e+03</td>
<td>5.79e+00</td>
<td>2.41e+02</td>
<td>2.87e+00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>max</td>
<td>5.77e+01</td>
<td>2.39e+02</td>
<td>4.01e+06</td>
<td>4.76e+02</td>
<td>8.28e+02</td>
<td>4.54e+00</td>
<td>2.06e+01</td>
<td>1.33e+03</td>
<td>5.45e+00</td>
<td>6.50e+00</td>
<td>4.43e+02</td>
<td>3.50e+00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mean</td>
<td>3.29e+01</td>
<td>2.15e+02</td>
<td>2.06e+06</td>
<td>1.31e+02</td>
<td>4.34e+02</td>
<td>9.36e+02</td>
<td>1.46e+00</td>
<td>2.05e+01</td>
<td>7.06e+04</td>
<td>2.46e+03</td>
<td>4.76e+00</td>
<td>1.94e+02</td>
<td>2.56e+00</td>
</tr>
<tr>
<td></td>
<td>std</td>
<td>1.40e+01</td>
<td>6.18e+01</td>
<td>1.03e+06</td>
<td>9.32e+01</td>
<td>1.27e+02</td>
<td>1.00e+03</td>
<td>1.77e+00</td>
<td>8.74e+02</td>
<td>2.88e+04</td>
<td>1.16e+03</td>
<td>1.19e+00</td>
<td>1.02e+02</td>
<td>4.38e+01</td>
</tr>
</tbody>
</table>

**FES** = Function Evaluation Steps
**Prob.** = Probability
**f1** to **f2** = Objective Function Values
### Table 12: Best objective function error values reached after $10^3$, $10^4$ and $10^5$ FES in dimension $n = 30$. See caption of Table 11 for details.

#### Table 12: Best objective function error values reached after $10^3$, $10^4$ and $10^5$ FES in dimension $n = 30$. See caption of Table 11 for details.
Table 13: Best objective function error values reached after $10^3$, $10^4$ and $10^5$ FES in dimension $n = 50$. See caption of Table 11 for details.
make.inc File

C make.inc File

1 # CMA-ES make include file#
2
3 # PLATFORM #PLATFORM#
4 # Set Platform ("win" for windows, otherwise empty)
5 PLAT =
6
7 # COMPILER AND COMPILER OPTIONS #COMPILER AND COMPILER OPTIONS#
8 FC = ifort
9 CFLAGS = -O0
10
11 # MATLAB #MATLAB#
12 # Declare if Matlab is present on your machine
13 # (0 == 'no', 1 == 'yes')
14 HASMAT = 1
15
16 # If Matlab is present, please specify the appropriate
17 # include paths, see README.txt ('Linking Matlab Libraries')
18 MATINC = -I/Applications/MATLAB_R2007b/extern/include
19 MATLIB = -L/Applications/MATLAB_R2007b/bin/maci
20
21 # LAPACK #LAPACK#
22 # Tell the linker where to find your LAPACK library if not
23 # already included in your PATH variable
24 LAPACKLIB =
25
26 # PRECISION #PRECISION#
27 # Set the precision that should be used for computation
28 # (Single Precision = 1, Double Precision = 2)
29 # Please do a make clean whenever you change the precision
30 PREC = 2
31
32 # MPI #MPI#
33 # If MPI is installed, set MPI = 1, otherwise MPI = 0
34 # Define where to find the MPI include files needed for
35 # compilation and tell the linker where to find the
36 # MPI-Libraries if they are not already included in your
37 # PATH variable
38 MPI = 1
39 MPIINC = -I/Users/paulb/openmpi/include
40 MPILIB = -L/Users/paulb/openmpi/lib
41
42 # ADDITIONALS #ADDITIONALS#
43 # You might want to include additional source files or
44 # directories where to look for them, e.g. if you want to use
45 # test functions, residing in a different directory
46 ADDSOURCES = pscma_example.f90
47 ADDPATH =
48
D  Example Fortran Program to Call PS-CMA-ES

```
PROGRAM pscma_example
! Include necessary modules
USE cmaes_param_mod
USE cmaesOpts_mod
USE CEC2005

IMPLICIT NONE
#ifdef _HAVE_MPI_
INCLUDE "mpif.h"
#endif

! Interface to check parameters passed to cmaesStart
INTERFACE cmaesStart
SUBROUTINE cmaesStart ( fitfun , xstart , insigma , varargin )
USE cmaes_param_mod
USE cmaes_mod , ONLY : input
IMPLICIT NONE
EXTERNAL
REAL(MK) , INTENT(in) :: fitfun
REAL(MK) , INTENT(in) :: xstart (:)
REAL(MK) , INTENT(in) :: insigma (:)
REAL(MK) , INTENT(in) , OPTIONAL :: varargin
END SUBROUTINE cmaesStart
END INTERFACE cmaesStart

! Declare variables: problem dimension, start position
! and initial step size
INTEGER :: ierr
INTEGER, PARAMETER :: d = 10
REAL(MK) , DIMENSION(d) :: xstart
REAL(MK) , DIMENSION(d) :: insigma

#ifdef _HAVE_MPI_
CALL MPI_INIT(ierr)
CALL MPI_COMM_RANK(MPI_COMM_WORLD , MY_RANK , ierr)
CALL MPI_COMM_SIZE(MPI_COMM_WORLD , NUM_CMA_RUNS , ierr)
#endif

! Initialize start position and initial step size
xstart = 0.0_MK
insigma = 0.5_MK

! CMA options
options%PopSize = 6
options%StopMaxFunEvals = d*1.E4
options%flgGenData = .TRUE.

! Define constrained search region
ALLOCATE(options%LBounds(d))
options%LBounds = -5.0_MK
ALLOCATE(options%UBounds(d))
options%UBounds = 5.0_MK
```
! PSO options
options%pscma = .TRUE.
options%psoFreq = 200
options%psoWeight = 0.7

! Start CMA-ES
CALL cmaesStart(F09_shifted_rastrigin, xstart, insigma)

#ifdef HAVE_MPI_
CALL MPI_FINALIZE(ierr)
#endif

END PROGRAM pscma_example
### Available (PS-)CMA-ES Options

<table>
<thead>
<tr>
<th>Name</th>
<th>Default Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stop Criteria</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StopFitness</td>
<td>-Inf</td>
<td>Stop if ( f(x) &lt; \text{StopFitness} )</td>
</tr>
<tr>
<td>StopMaxFunEvals</td>
<td>Inf</td>
<td>Maximal number of FES</td>
</tr>
<tr>
<td>StopMaxIter</td>
<td>( \frac{1 \times 3(n+5)^2}{\sqrt{\text{PopSize}}} )</td>
<td>Maximal number of iterations</td>
</tr>
<tr>
<td>StopFunEvals</td>
<td>Inf</td>
<td>Stop after resp. evaluation</td>
</tr>
<tr>
<td>StopIter</td>
<td>Inf</td>
<td>Stop after resp. iteration</td>
</tr>
<tr>
<td>StopTo1X</td>
<td>( 1 \times 11 \times \max(\sigma_{\text{initial}}) )</td>
<td>Stop if x-change &lt; \text{StopTo1X}</td>
</tr>
<tr>
<td>StopTo1UpX</td>
<td>( 1 \times 3 \times \max(\sigma_{\text{initial}}) )</td>
<td>Stop if x-change &gt; \text{StopTo1UpX}</td>
</tr>
<tr>
<td>StopTo1Fun</td>
<td>( 1 \times 12 )</td>
<td>Stop if fun-change &lt; \text{StopTo1Fun}</td>
</tr>
<tr>
<td>StopTo1HistFun</td>
<td>( 1 \times 13 )</td>
<td>Stop if back fun-change &lt; \text{StopTo1Fun}</td>
</tr>
<tr>
<td>StopOnWarnings</td>
<td>true</td>
<td>self-explanatory</td>
</tr>
<tr>
<td><strong>CMA-ES Strat. Param.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PopSize</td>
<td>( 4 + \lfloor 3 \ln(n) \rfloor )</td>
<td>Population Size ( \lambda )</td>
</tr>
<tr>
<td>ParentNumber</td>
<td>( \lfloor \frac{\lambda}{2} \rfloor )</td>
<td>Parent Number ( \mu )</td>
</tr>
<tr>
<td>RecombinationWeights</td>
<td>3</td>
<td>Super-linear (3), linear (2) or equal (1)</td>
</tr>
<tr>
<td>DiffMaxChange</td>
<td>Inf</td>
<td>Maximal variable change</td>
</tr>
<tr>
<td>DiffMinChange</td>
<td>0</td>
<td>Minimal variable change</td>
</tr>
<tr>
<td>LBounds</td>
<td>-Inf</td>
<td>Lower bounds</td>
</tr>
<tr>
<td>UBounds</td>
<td>Inf</td>
<td>Upper bounds</td>
</tr>
<tr>
<td><strong>Particle Swarm Opts.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pscma</td>
<td>false</td>
<td>Switch PS-CMA-ES on or off</td>
</tr>
<tr>
<td>psoWeight</td>
<td>0.7</td>
<td>( c ), see Equation (4.1)</td>
</tr>
<tr>
<td>psoFreq</td>
<td>200</td>
<td>Frequency of PSO updates</td>
</tr>
<tr>
<td><strong>Benchmark Opts.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>benchmark</td>
<td>false</td>
<td>Switch Benchmark on or off</td>
</tr>
<tr>
<td>global_min</td>
<td>0.0</td>
<td>Global minimum</td>
</tr>
<tr>
<td>accuracy</td>
<td>0.0</td>
<td>Successful run if (</td>
</tr>
<tr>
<td><strong>Others</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EvalInitialX</td>
<td>true</td>
<td>Evaluate initial solution</td>
</tr>
<tr>
<td>WarnOnEqualFunctionValues</td>
<td>true</td>
<td>Self-explanatory</td>
</tr>
<tr>
<td>funcName</td>
<td></td>
<td>Objective function name</td>
</tr>
<tr>
<td>flgGenData</td>
<td>true</td>
<td>Save data of all iterations</td>
</tr>
<tr>
<td>VerboseModulo</td>
<td>100</td>
<td>Messaging after every i-th iteration</td>
</tr>
</tbody>
</table>

Table 14: Available (PS-)CMA-ES Options. All options belong to the derived data type options. For example, options\%pscma = .TRUE. turns on the particle swarm feature.
Figure 15: PS-CMA-ES (swarm size = 3, update frequency = 10) and CMA-ES Evolution on Schwefel’s Problem. Displayed are candidate solutions, the Covariance Matrix Error Ellipse and the Current Global Best Position (*).
Figure 15: Continued.
Figure 15: Continued. (k) and (l) show all Sample Points during Evolution for CMA-ES and PS-CMA-ES.
References


[54] Optimization Toolbox.  


[57] Lawrence Livermore National Laboratory. Message Passing Interface (MPI).  
