Impacts of Invariance in Search: When CMA-ES and PSO Face Ill-Conditioned and Non-Separable Problems

Nikolaus Hansen, Raymond Ros, Nikolas Mauny, Marc Schoenauer, Anne Auger

Impacts of Invariance in Search: When CMA-ES and PSO Face Ill-Conditioned and Non-Separable Problems

Nikolaus Hansen, Raymond Ros, Nikolas Mauny, Marc Schoenauer, Anne Auger

Nikolaus Hansen[1] is with Microsoft Research-INRIA Joint Centre, Parc Orsay Université, 28, rue Jean Rostand, 91893 Orsay Cedex France. Raymond Ros is with LRI, Project-team TAO, Université Paris-Sud, 91405 Orsay Cedex, France. All other authors are with INRIA Saclay, Project-team TAO, LRI, Université Paris-Sud, 91405 Orsay Cedex, France. For 1st, 4th and 5th author mail to `forename.name@inria.fr`*, for 2nd and 3rd author mail to* `forename.name@lri.fr`*.*

Abstract

This paper investigates the behavior of PSO (particle swarm optimization) and CMA-ES (covariance matrix adaptation evolution strategy) on ill-conditioned functions. The paper also highlights momentum as important common concept used in both algorithms and reviews important invariance properties. On *separable*, ill-conditioned functions, PSO performs very well and outperforms CMA-ES by a factor of up to five. On the same but rotated functions, the performance of CMA-ES is unchanged, while the performance of PSO declines dramatically: on *non-separable*, ill-conditioned functions we find the search costs (number of function evaluations) of PSO increasing roughly proportional with the condition number and CMA-ES outperforms PSO by orders of magnitude. The strong dependency of PSO on rotations originates from random events that are only independent within the given coordinate system. The CMA-ES adapts the coordinate system where the independent events take place and is rotational invariant. We argue that invariance properties, like rotational invariance, are desirable, because they increase the predictive power of performance results by inducing problem equivalence classes.

Key words: Particle Swarm Optimization, PSO, Covariance Matrix Adaptation, Evolution Strategy, CMA-ES, performance assessment, ill-conditioned problems, non-separable problems, invariance

1. Introduction

In this paper we consider stochastic methods that aim to minimize a non-linear objective function

$$f : X \subset \mathbb{R}^n \to \mathbb{R} \qquad (1)$$
$$\boldsymbol{x} \mapsto f(\boldsymbol{x}) \ ,$$

where the search space X is typically a convex subset of $\mathbb{R}^n$. Throughout this paper, X equals $\mathbb{R}^n$. The typical search space dimensionality n, the number of design variables, ranges between two and a few hundred. In this paper, the search *costs* are defined as the number of function evaluations on f. The search objectives are twofold: finding 1) a good solution $\boldsymbol{x}$, where $f(\boldsymbol{x})$ is as small as possible, while 2) the search costs should be as small as possible. Our preferred performance measure will be the expected costs needed to fall below a target function value.

1.1. Why Search is Difficult

In order to understand success and failure of search algorithms we need to understand the difficulties of the objective function f. Understanding f also leads to criteria on how to categorize and design test functions in a useful way. In the following, we consider two function properties that make a problem difficult: ill-conditioning, and, as a prerequisite, non-separability.

1.1.1. Decomposability and Separability

We call an objective function, $f : \boldsymbol{x} \mapsto f(\boldsymbol{x})$, *separable with respect to coordinate i*, if the optimal value for the i-th coordinate x_i does not depend on the choice of the remaining coordinates. We call the objective function *separable*, if it is separable with respect to each coordinate. More formally, let $\boldsymbol{e}_i \in \mathbb{R}^n$ be the i-th unit vector. A function is said separable with respect to coordinate i if, for all $\boldsymbol{y}, \boldsymbol{z} \in \mathbb{R}^n$,

$$\arg\min_{\lambda \in \mathbb{R}} f(\boldsymbol{y} - y_i \boldsymbol{e}_i + \lambda \boldsymbol{e}_i) = \arg\min_{\lambda \in \mathbb{R}} f(\boldsymbol{z} - z_i \boldsymbol{e}_i + \lambda \boldsymbol{e}_i) \ .$$

Many well-known test functions are additively decomposable. They can be written as a sum of n one-dimensional functions f_i like $f(\boldsymbol{x}) = \sum_{i=1}^n f_i(x_i)$. Additively decomposable functions are separable while separable functions are not necessarily additively decomposable.

Separable functions are not subject to the curse of dimensionality. The global function minimizer can be found by minimizing n one-dimensional objective functions $f_i : \lambda \in \mathbb{R} \mapsto f(\boldsymbol{x} + (\lambda - x_i)\boldsymbol{e}_i)$, $i = 1, \ldots, n$, for any given $\boldsymbol{x}$ in $\mathbb{R}^n$. Their difficulty scales linearly with the search space dimension. In contrast, the search space volume increases exponentially fast with the dimension, leading to the notion of curse of dimensionality. In this paper we will conduct experiments on separable ("easy") and non-separable ("hard") functions.

1.1.2. Ill-Conditioning

Informally speaking, the term ill-conditioned refers to a situation where different variables, or different directions in search space, show a largely different sensitivity in their contribution to the objective function value.

For a convex-quadratic function $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x}$, where $\boldsymbol{H}$ is symmetric positive definite, the problem condition is well-defined by the condition number of the Hessian matrix $\boldsymbol{H}$. The condition number of $\boldsymbol{H}$ is the ratio between its largest and smallest eigenvalue. Figuratively, the eigenvalues correspond to the squared relative lengths of the principal axes of the ellipsoid $\{\boldsymbol{x} \,|\, \boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} = 1\}$. For points located on the principal axes, the gradient aligns with the axis direction, and the axis length determines the magnitude of movement that would be needed to achieve a certain change in function value.

More generally, we can call a function ill-conditioned if for points with similar function values the minimal displacement (in search space) that produces a given function value improvement differs by orders of magnitude.

In practice, little domain knowledge is usually sufficient to identify separable problems. Therefore, n-dimensional search problems often exhibit intricate dependencies between their design variables and these ill-conditioned problems often lead to premature convergence.

1.2. Objective of this Paper

This article aims to quantify performance depending on ill-conditioning for two stochastic search methods. The two considered methods are Particle Swarm Optimization (PSO) [1, 2, 3, 4] and Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [5, 6, 7]. We will answer the following questions *quantitatively*.

1. How well do PSO and CMA-ES perform on ill-conditioned problems?
2. How strongly does the performance depend on coordinate system rotations?

To this end, we investigate PSO and CMA-ES on a small number of carefully chosen test functions *with varying condition number*.

This paper neither introduces a new method nor improves a known one. The purpose is, on the one hand, to gain specific and important insight into *well-established* methods, and, on the other hand, to introduce useful and powerful evaluation methodologies in the domain of evolutionary and swarm optimization encouraging their usage in further empirical studies.

The next section reviews the concept of invariance in search and optimization. The following two sections review PSO and CMA-ES in turn. Section 3.3 describes a common concept that is extensively utilized in both algorithms. Section 4 describes the experimental set-up and Section 5 presents the experimental results. Summary and conclusion are provided in Section 7.

2. Invariance

Invariance, in physics referred to as *symmetry*, is a fundamental concept in science. The purpose of invariance is well reflected in the following quote attributed to Albert Einstein: *The grand aim of all science is to cover the greatest number of empirical facts by logical deduction from the smallest number of hypotheses or axioms.* Invariance is the mathematical concept associated with this aim. For example, we desire a physical law or biological model to be invariant to environmental parameters, say weekday, temperature, or air humidity. Inclusion of these parameters into the model or the need for controlling them makes the model more complex and/or less general. The more invariance properties a model exhibits, or the fewer dependencies on exogenous parameters the model reveals, the wider is its applicability and the greater is its predictive power.

The same idea holds for invariance properties of search algorithms. In search, invariance properties induce equivalence classes of objective functions, on which the performance of the search algorithm is identical. Consequently, any result observed on a real world problem, or on a test function, does not only hold for this single problem instance, but inevitably *generalizes to the complete class of problems induced by the invariance property*, that the tested problem is an element of. Hence stronger statements on the performance of the search algorithm can be made—a greater number of empirical facts is covered.[2]

The drawback to invariance properties in search is that some information cannot be exploited anymore. For example, rotational invariance means to abandon exploitation of the orientation of the given coordinate system and therefore exploitation of separability. We review important invariance properties of search algorithms after giving a formal definition.

Definition 1 (Invariance). *Let $\mathcal{T} \subset \{T : \mathbb{R} \to \mathbb{R}\}$, $\mathcal{U} \subset \{U : \mathbb{R}^n \to \mathbb{R}^n\}$, where $\mathcal{T}$ and $\mathcal{U}$ contain the identity. Let S be the state space of the search algorithm, $s \in S$ and $\mathcal{A}_f : S \to S$ an iteration step of the algorithm under objective function f. The algorithm $\mathcal{A}$ **is invariant under $\mathcal{T}$ and $\mathcal{U}$** if for all $T \in \mathcal{T}$, $U \in \mathcal{U}$ there exists a bijective state space transformation $J_{T,U} : S \to S$ such that for all $f \in \mathbb{R}^n \to \mathbb{R}$, $s \in S$*

$$J_{T,U} \circ \mathcal{A}_{T\circ f\circ U}(s) = \mathcal{A}_f \circ J_{T,U}(s) \ . \tag{2}$$

or equivalently

$$\mathcal{A}_{T\circ f\circ U}(s) = J_{T,U}^{-1} \circ \mathcal{A}_f \circ J_{T,U}(s) \ , \tag{3}$$

For randomized algorithms, the equalities hold almost surely, given appropriately coupled random number realizations, otherwise in distribution. The set of functions $\{T \circ f \circ U \mid T \in \mathcal{T},\ U \in \mathcal{U}\}$ is an invariance set of f for algorithm $\mathcal{A}$.

Equation (3) implies (trivially) for all $k \in \mathbb{N}$ that

$$\mathcal{A}^k_{T\circ f\circ U}(s) = J_{T,U}^{-1} \circ \mathcal{A}^k_f \circ J_{T,U}(s) \ , \tag{4}$$

where $\mathcal{A}^k(s)$ denotes k iteration steps of the algorithm starting from s. Equation (4) illustrates that for any $f : \mathbb{R}^n \to \mathbb{R}$, $T \in \mathcal{T}$, $U \in \mathcal{U}$, the algorithm $\mathcal{A}$ optimizes the function $T \circ f \circ$

[2] Invariance does *not* imply good performance per se, it only provides the means *to generalize* (favorable and unfavorable) performance observations.

U with initial state s just like the function f with initial state $J_{T,U}(s)$. We will see that in some cases $J_{T,U}$ is the identity. Otherwise, finding the "right" $J_{T,U}(s)$ can be non-trivial and an adaptation process must move s to $J_{T,U}(s)$, such that *after an adaptation phase* any function $T \circ f \circ U$ is optimized just like the function f.[3] This is particularly attractive, if f is the easiest function in the invariance set. Any invariance set of functions guaranties the existence of a, possibly larger, set of functions for which the invariance property holds and that also defines an *equivalence class* in the set of all functions. Therefore the invariance properties discussed in the following indeed define equivalence classes.

By reducing $\mathcal{U}$ to the identity transformation, one can consider invariance under a set $\mathcal{T} \subset \{T : \mathbb{R} \to \mathbb{R}\}$ solely. We will talk about *invariance under function value transformations*. Similarly, reducing $\mathcal{T}$ to the identity transformation, one can consider invariance under $\mathcal{U} \subset \{U : \mathbb{R}^n \to \mathbb{R}^n\}$ solely. We will talk about *invariance under search space transformations.*

2.1. Invariance under Function Value Transformations

We detail now different invariance properties under transformations T of the objective function value. This means equivalence of two functions f and $T \circ f$, where T belongs to $\mathcal{T}$ as given below.

- Invariance to adding a constant to the function value, that is under the set of transformations $\mathcal{T} = \{T : x \in \mathbb{R} \to x + a, a \in \mathbb{R}\}$.
- Invariance under scaling of the function value, that is under the set of transformations $\mathcal{T} = \{T : x \in \mathbb{R} \to a \times x, a > 0\}$.
- Invariance under order preserving transformations of the objective function value, that is under the set $\mathcal{T}$ of strictly monotonically increasing functions. Invariance under order preserving transformations includes both above given invariance properties and is much more general.

Because PSO and CMA-ES depend only on a *ranking* of function values, they achieve the above listed invariance properties. The sequence of generated search points is independent of T. We believe that this is a very important feature of comparison-based search methods [8].

2.2. Invariance under Search Space Transformations

We describe now invariance properties under different search space transformations U, that is, equivalence of two functions f and $f \circ U$, where $U : \mathbb{R}^n \to \mathbb{R}$ belongs to a set of search space transformations $\mathcal{U}$ as given below. Strictly speaking, invariance under U only holds, if also the *initial conditions* are chosen appropriately, i.e., $J_{T,U}$ is in general not equal to the identity. In PSO, the setting of the initial swarm and the initial velocities must be chosen accordingly, in CMA-ES initial mean and covariance matrix of the search distribution.

[3] Note that the last step $J_{T,U}^{-1}$ is actually not needed, because the algorithm actually runs on $T \circ f \circ U$ and delivers the desired solution for this function, not for f.

Translation invariance means invariance under $\mathcal{U} = \{U : \boldsymbol{x} \mapsto \boldsymbol{x} + \boldsymbol{a} \text{ for } \boldsymbol{a} \in \mathbb{R}^n\}$. Translation invariance must be taken for granted in continuous domain search. Lacking translational invariance must be interpreted as having an *inherent*, problem independent assumption about the location of the optimal solution. For a search algorithm this seems to be a contradiction in terms. For example, if zero is a distinguished solution point for the algorithm, on many test functions exceptional performance can be achieved, but the results are entirely artificial. However, the initial solution should be considered as a *justified*, problem dependent best first guess about the location of the optimal solution that can (and should) be exploited by the algorithm.

Scale invariance means invariance under $\mathcal{U} = \{U : \boldsymbol{x} \mapsto \alpha\boldsymbol{x} \text{ for } \alpha > 0\}$. From the algorithm descriptions as given below, and given the appropriate initial conditions, one can easily verify that PSO and CMA-ES are scale invariant.

Finally, we have invariance properties where $U(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$ and $\boldsymbol{A}$ is a full rank matrix.

Diagonal invariance is invariance under diagonal linear transformations, *i.e.* under a scaling of variables. The matrix $\boldsymbol{A}$ is diagonal. Again, one can easily verify that PSO is invariant under diagonal linear transformations (just as for scale invariance, as PSO is defined coordinate-wise).

Rotational invariance is invariance under angle preserving, *i.e.* rigid linear transformations of the search space (rotation, reflection). That is, $\boldsymbol{A}$ is an orthogonal matrix. Rotational invariance is closely related to decomposability and separability (see above), because, in most cases, a separable function becomes non-separable under rotation. Therefore a search algorithm can only *either* exploit separability *or* be rotational invariant.

General linear invariance is invariance under any full rank, *i.e.* invertible matrix $\boldsymbol{A}$. This invariance includes rotational and diagonal invariance and requires to abandon any inherent model of isotropy and scales. The CMA-ES is invariant under general linear transformations [5].

Rotational invariance and general linear invariance of PSO depend on the way the update equations are implemented and will be discussed below. We conjecture that the impact of an invariance property is related to the degrees of freedom related to the transformation. Consequently, orthogonal and general linear invariance must be considered important just like invariance under order preserving transformations of the objective function value.

In practice, initial conditions often cannot be chosen accordingly to the desired search space invariance property. Therefore, an invariant search algorithm must also be adaptive: the initial state must evolve into "the invariant state" ($J_{T,U}(s)$ in Equation (4)), rendering the algorithm as independent as possible of the initial conditions. Adaptivity has the additional advantage that the state can be adapted to the actual position in

search space.[4] While any constant sample distribution exhibits general linear invariance, *given the initial distribution is transformed accordingly*, this invariance alone is rather useless—adaptivity is the essential counterpart of any invariance that depends on the initial conditions.

3. Two Stochastic Search Methods

We investigate two well-established algorithms from two different domains in bio-inspired stochastic search, Particle Swarm Optimization and Evolution Strategies. We have chosen two comparatively stable and well-recognized instances and implementations for each of them.

3.1. Particle Swarm Optimization (PSO)

Particle Swarm Optimization [PSO, 1, 2, 3, 4] is a stochastic search algorithm, inspired by flock behavior, that has become increasingly popular in the recent years. In PSO, a number of particles evolve their positions in the search space over time. Here, we confine ourselves to one particular instance of PSO, the so-called *standard PSO 2006*, referred to as s-PSO in the remainder. The following description of s-PSO is based on its implementation in C, as used in the paper and given elsewhere[5]. We first describe s-PSO using a non-standard notation (without velocities) thereby gaining different insights.

The s-PSO tracks a number of particles (solution vectors) in a swarm. The default swarm size of s-PSO is $S = 10+\lfloor 2\sqrt{n} \rfloor$. For each particle, $\boldsymbol{x} \in \mathbb{R}^n$, its *previous best position* $\boldsymbol{p}$ is recorded. Let $t \in \mathbb{N}$ be the time index and $\boldsymbol{x}^t$ the position of particle $\boldsymbol{x}$ at time t, then we have

$$\boldsymbol{p} \in \{\boldsymbol{x}^t, \boldsymbol{x}^{t-1}, \ldots, \boldsymbol{x}^1\} \tag{5}$$

chosen such that $f(\boldsymbol{p})$ is minimal. Additionally, each particle serves as informant based on a neighborhood topology. Each time the overall best particle does not improve, the neighborhood topology is recomputed: the particles for which a particle serves as informant are randomly drawn from all particles with replacement in $K = 3$ rounds for each particle anew. Therefore, each particle serves as informant for between zero and K other particles. Additionally, a particle always informs itself. Therefore, in the extreme cases, a particle can only have itself as informant, or the complete swarm.[6]

The informants serve to compute the "global best" position $\boldsymbol{g}$ of the particle, that is the best previous-best position of its current informants, including itself.[7] If all particles are informants, $\boldsymbol{g}$ is the overall (globally) best position ever visited.

In order to compute, for each particle $\boldsymbol{x}$, the new position, $\boldsymbol{x}^{t+1}$, the four vectors $\boldsymbol{x}^t$, $\boldsymbol{x}^{t-1}$, $\boldsymbol{p}$, and $\boldsymbol{g}$ are used. The new position is coordinate wise a linear combination of these four vectors with coefficients summing to one. At each iteration step t for each particle $\boldsymbol{x}^t = (x_i^t)_{i=1,\ldots,n}$ a new position $\boldsymbol{x}^{t+1}$ is computed. We first consider the *stochastic* part of this computation which leads to the intermediate position

$$\begin{aligned} x_i^{t+\frac{1}{2}} &= x_i^t + U_i^+ \left(p_i - x_i^t\right) + V_i^+ \left(g_i - x_i^t\right) \\ &= (1 - U_i^+ - V_i^+)\, x_i^t + U_i^+ p_i + V_i^+ g_i \\ &= \frac{p_i + g_i}{2} + (p_i - x_i^t)U_i + (g_i - x_i^t)V_i \end{aligned} \tag{6}$$

for each coordinate $i = 1, \ldots, n$, where U_i^+ and V_i^+ are uniformly distributed in $[0, \varphi]$ with $\varphi = \log(2) + \frac{1}{2} \approx 1.19$, and $U_i = U_i^+ - \frac{1}{2}$, $V_i = V_i^+ - \frac{1}{2} \in [-0.5, 0.7]$. The final new position is a *deterministic* shift of $\boldsymbol{x}^{t+\frac{1}{2}}$ according to

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^{t+\frac{1}{2}} + w\left(\boldsymbol{x}^t - \boldsymbol{x}^{t-1}\right) \tag{7}$$

where the inertia weight w equals to $\frac{1}{2\log(2)} \approx 0.72$. Equation 7 implements a momentum by repeating the position change of the last iteration step in an exponential lossy way. The entire step taken by the particle is also called *velocity*

$$\begin{aligned} \boldsymbol{v}^{t+1} &= \boldsymbol{x}^{t+1} - \boldsymbol{x}^t \\ &= \boldsymbol{x}^{t+\frac{1}{2}} + w\left(\boldsymbol{x}^t - \boldsymbol{x}^{t-1}\right) - \boldsymbol{x}^t \\ &= w\left(\boldsymbol{x}^t - \boldsymbol{x}^{t-1}\right) + (\boldsymbol{x}^{t+\frac{1}{2}} - \boldsymbol{x}^t) \\ &= w\,\boldsymbol{v}^t + (\boldsymbol{x}^{t+\frac{1}{2}} - \boldsymbol{x}^t) \ . \end{aligned} \tag{8}$$

3.1.1. Interpretations

The typical interpretation of Equation (6) is that the particle is pulled toward the best positions $\boldsymbol{g}$ and $\boldsymbol{p}$ as it can be implied from the first line of Equation (6). Because φ is greater than one, in fact, the particle can be pulled beyond the best positions.

A second interpretation is that Equation (6) resembles a recombination operator of real-coded genetic algorithms [9]. The new position results from a linear combination of x_i^t, p_i, and g_i, where the coefficients, different in each coordinate i, sum to one. This idea suggests itself from the second line of Equation (6). Realizing that the expected value for the coefficient in front of x_i^t is smaller than zero makes this interpretation less attractive. When including Equation (7) into this view point, the four values x_i^{t-1}, x_i^t, p_i, and g_i are linearly combined with a negative coefficient for x_i^{t-1}.

A third interpretation of Equation (6), emphasized in the last line, is that the new position is sampled around the average of $\boldsymbol{g}$ and $\boldsymbol{p}$. A similar, even more general reformulation was analyzed in [4]. According to the third interpretation, $\boldsymbol{g}$ and $\boldsymbol{p}$ are averaged, and $\boldsymbol{x}^t$ primarily affects the perturbation width around this average, according to its coordinate-wise distance to $\boldsymbol{g}$ and $\boldsymbol{p}$. The width is chosen such that the particle $\boldsymbol{x}^t$ itself is not outside the sampling domain. If both, p_i and g_i, are either smaller than x_i or larger than x_i, the latter defines the domain boundary in coordinate i. Because $\boldsymbol{x}^t$ is located on the boundary of the

[4] The most prominent example are step-sizes and velocities: internal state variables that should ideally decrease by orders of magnitude when the optimal solution is approached.

[5] The C code of the "Standard PSO 2006" can be found in `http://www.particleswarm.info/Standard_PSO_2006.c`.

[6] The probability for a particle to only have itself as informant is $\left(\frac{S-1}{S}\right)^{K(S-1)}$, where S is the swarm size. The probability to have all particles as informant is $\left(1 - \left(\frac{S-1}{S}\right)^K\right)^{S-1} \leq \left(\frac{K}{S}\right)^{S-1}$.

[7] Notice that $f(\boldsymbol{g}) \leq f(\boldsymbol{p})$ and $Pr(\boldsymbol{g} = \boldsymbol{p}) > \frac{1}{S}$. Also $f(\boldsymbol{g})$ can increase, *i.e.* $\boldsymbol{g}$ can become worse from one iteration step to the next, because the informants can change from one generation to the next. In contrast $f(\boldsymbol{p})$ is non-increasing.

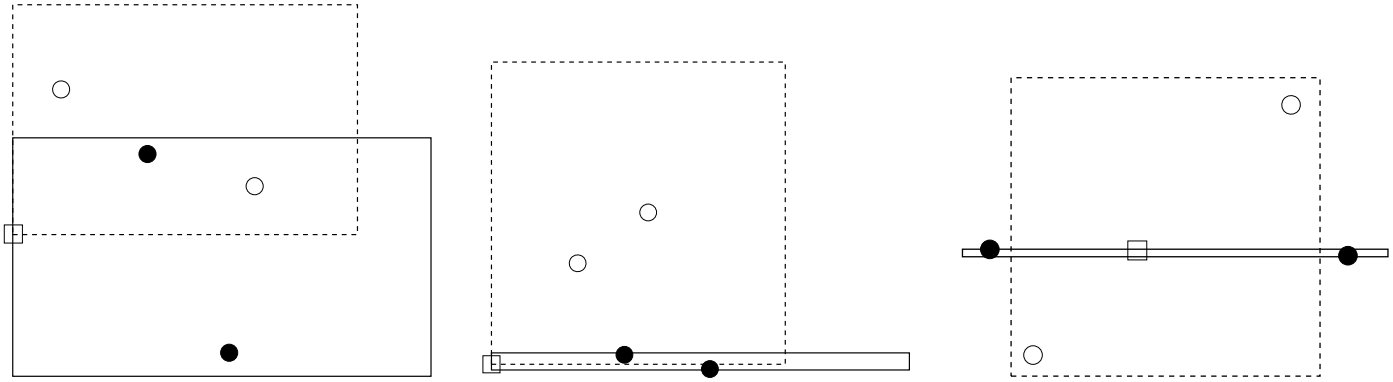

Figure 1: Domain of non-zero density for the intermediate particle position $\boldsymbol{x}^{t+\frac{1}{2}} = \boldsymbol{x}^{t+1} - w(\boldsymbol{x}^t - \boldsymbol{x}^{t-1})$, in three cases (solid rectangles), and after a 45 ° rotation around □ (dashed rectangles). The symbols indicate: □ = particle $\boldsymbol{x}^t$, ● = $\boldsymbol{p}$ and $\boldsymbol{g}$ (interchangeably), ◯ = $\boldsymbol{p}$ and $\boldsymbol{g}$ in the rotated case.

sample domain if at least one of its coordinates is on the boundary, we reckon, from a simple combinatorial reasoning, that $\boldsymbol{x}^t$ is on the boundary with probability larger than $1 - \frac{1}{2}^n$. **Figure 1** illustrates this view point, as in the left and middle subfigures the particle (□) lies on the boundary. Because $\varphi \approx 1.19 > 1$, the sample width is about a factor of $\frac{\varphi - 0.5}{0.5} \approx 1.4$ times larger opposite to $\boldsymbol{x}^t$ than toward $\boldsymbol{x}^t$.

The effect of a search space rotation on Equation (6) is illustrated in **Figure 1**. The effect can be dramatic. The distribution becomes narrow only if $\boldsymbol{g}$, $\boldsymbol{p}$, and $\boldsymbol{x}^t$ are all aligned with a coordinate axis. A narrow sampling along, for example, a diagonal cannot be realized. A swarm, intentionally placed on a diagonal, resembles a quadratic shape after the next iteration. A swarm, intentionally placed on a coordinate axis remains on this axis forever. The illustration suggests a significant dependency on rotations of the underlying problem. This dependency will later be empirically quantified.

3.1.2. Preserving Rotational Invariance

Rotational invariance of s-PSO, and general linear invariance, depend on a subtlety in Equation (6), namely whether the random variable realizations for U_i and V_i are independent or identical for all $i = 1, \ldots, n$. While this subtlety is often left unspecified, in s-PSO different realizations are used. Only if identical realizations are used for all i the algorithm becomes invariant under linear transformations of the search space [10]. In this case the trajectory of each particle tends to collapse into a "long narrow plane", reminiscent of a line search, leading to unfavorable performance [11, 12]. This behavior is in accordance with our experience and understanding that any search algorithm with general linear invariance tends to collapse (degenerate) into a low-dimensional subspace. This is the case for CMA-ES: if the learning rate for the covariance matrix is chosen too large, the covariance matrix degenerates such that the smallest eigenvalue converges faster to zero than the largest. This is the case for the Nelder-Mead algorithm: with increasing search space dimension the simplex has an increasing tendency to collapse.

This phenomenon might be denoted as *invariance-diversity dilemma*. Evolutionary algorithms sometimes use a large population size to combat this degeneration, implying large search costs. In contrast, *rotationally* invariant algorithms do not necessarily degenerate. For example, any isotropic perturbation prevents effectively the degeneration into low-dimensional subspaces and is a rotationally invariant operator.

In [10], a modified linear update is proposed that preserves rotational invariance, in their terminology *frame invariance*. Despite that their *"goal is not to propose yet another competitive and/or superior PSO variant, but merely to illustrate that formulations that are both diverse and invariant do exist"* [10], the modification outperforms the original algorithm on a convex-quadratic function with moderate condition number by a factor of about three, and on a variant of the Rosenbrock function (where no such factor can be concluded from the presented data). In order to maintain diversity, rotation matrices with small angles are applied to the difference vectors. The angle parameter determines the mean width of the rotation and controls the compromise between diversity preservation (for larger values) versus direction preservation (for smaller values). General linear invariance is not preserved. Therefore, the "amount of elongation" that a swarm can exhibit will be limited, depending on the angle parameter chosen.

3.2. Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

Evolutionary Algorithms are stochastic search methods inspired by the principles of biological evolution. Similar to PSO, they operate on a set of search points. Evolution Strategies [ESs, 13, 14, 15] are a class of evolutionary algorithms typically using a multi-variate normal mutation distribution. Here, we confine ourselves to the Covariance Matrix Adaptation Evolution Strategy [CMA-ES, 5, 7, 16], considered as state-of-the-art in continuous domain evolutionary computation [17]. We give a brief description of the $(\mu/\mu_\mathrm{w}, \lambda)$-CMA-ES as found in [6] and used for our experiments.

In the CMA-ES, in each iteration step t, new individuals $\boldsymbol{x}_i \in \mathbb{R}^n$ are generated by sampling a multi-variate normal distribution,

$$\boldsymbol{x}_i = \boldsymbol{m}^t + \sigma^t \times \mathcal{N}_i\left(\boldsymbol{0}, \boldsymbol{C}^t\right) \qquad \text{for } i = 1, \ldots, \lambda \ , \qquad (9)$$

where $\mathcal{N}(\mathbf{0}, \boldsymbol{C}^t)$ is a normal distribution with mean $\mathbf{0}$ and $n \times n$ covariance matrix $\boldsymbol{C}^t$, and $\sigma^t > 0$ is the step-size. All individuals obey the same distribution with mean $\boldsymbol{m}^t$. After $\lambda = 4 + \lfloor 3 \ln n \rfloor$ individuals have been sampled, evaluated on f, and sorted according to their objective function values, the distribution parameters $\boldsymbol{m}^t$, σ^t, and $\boldsymbol{C}^t$ are updated for a new iteration step using the sorted population. The new mean $\boldsymbol{m}^{t+1}$ obeys

$$\boldsymbol{m}^{t+1} = \sum_{i=1}^{\mu} w_i \, \boldsymbol{x}_{i_f} \ , \tag{10}$$

where $\mu = \left\lfloor \frac{\lambda}{2} \right\rfloor$, $w_i = \frac{\ln(\mu+1) - \ln i}{\sum_{j=1}^{\mu} (\ln(\mu+1) - \ln j)}$ are the recombination weights and i_f denotes the index of the i-th best individual. Two so-called evolution paths are computed. They track the history of changes of the population mean with an exponential decay of the past.

$$\boldsymbol{p}_\sigma^{t+1} = (1 - c_\sigma) \, \boldsymbol{p}_\sigma^t + \sqrt{c_\sigma(2 - c_\sigma)\mu_\mathrm{w}} \, \frac{1}{\sigma^t} \, (\boldsymbol{C}^t)^{-\frac{1}{2}} (\boldsymbol{m}^{t+1} - \boldsymbol{m}^t) \tag{11}$$

$$\boldsymbol{p}_\mathrm{c}^{t+1} = (1 - c_\mathrm{c}) \, \boldsymbol{p}_\mathrm{c}^t + h_\sigma \sqrt{c_\mathrm{c}(2 - c_\mathrm{c})\mu_\mathrm{w}} \, \frac{1}{\sigma^t} \, (\boldsymbol{m}^{t+1} - \boldsymbol{m}^t) \ , \tag{12}$$

where $(\boldsymbol{C}^t)^{-\frac{1}{2}}$ is symmetric, positive and satisfies $(\boldsymbol{C}^t)^{-\frac{1}{2}} (\boldsymbol{C}^t)^{-\frac{1}{2}} = (\boldsymbol{C}^t)^{-1}$, $\mu_\mathrm{w}^{-1} = \sum_{i=1}^{\mu} w_i^2$, the cumulation parameters are chosen as $c_\sigma = \frac{\mu_\mathrm{w}+2}{n+\mu_\mathrm{w}+3}$, $c_\mathrm{c} = \frac{4}{n+4}$ and h_σ takes usually the value one and becomes zero if $\|\boldsymbol{p}_\sigma^{t+1}\|$ is large, as given in [6]. Finally, the evolution paths are used to update the step-size and the covariance matrix,

$$\sigma^{t+1} = \sigma^t \times \exp\left(\frac{c_\sigma}{d_\sigma} \left(\frac{\|\boldsymbol{p}_\sigma^{t+1}\|}{\mathsf{E}\|\mathcal{N}(\mathbf{0}, \boldsymbol{I})\|} - 1 \right) \right) \tag{13}$$

$$\boldsymbol{C}^{t+1} = (1 - c_1 - c_\mu) \, \boldsymbol{C}^t + c_1 \boldsymbol{p}_\mathrm{c}^{t+1} \boldsymbol{p}_\mathrm{c}^{t+1\,T} + c_\mu \sum_{i=1}^{\mu} w_i \, \frac{\boldsymbol{x}_{i_f} - \boldsymbol{m}^t}{\sigma^t} \cdot \frac{(\boldsymbol{x}_{i_f} - \boldsymbol{m}^t)^T}{\sigma^t} \ , \tag{14}$$

where $d_\sigma = 1 + c_\sigma + 2 \, \max\left(0, \, \sqrt{\frac{\mu_\mathrm{w}-1}{n+1}} - 1\right)$ is usually close to one, and the learning rates for the covariance matrix obey

$$c_1 = \frac{1}{\mu_\mathrm{w}} \left(\left(1 - \frac{1}{\mu_\mathrm{w}}\right) \min\left(1, \frac{2\mu_\mathrm{w} - 1}{(n+2)^2 + \mu_\mathrm{w}}\right) + \frac{1}{\mu_\mathrm{w}} \frac{2}{(n + \sqrt{2})^2} \right) \tag{15}$$

$$c_\mu = (\mu_\mathrm{w} - 1) \, c_1 \ . \tag{16}$$

All parameters are taken from [6].

The CMA-ES possesses all invariance properties discussed in Sect. 2 (while the initial values of $\boldsymbol{m}$ and $\boldsymbol{C}$ must be adjusted accordingly). The population size λ is the only strategy internal parameter that needs to be adjusted depending on the objective function. Yet, a default choice, as used in this paper, and a logical variation procedure within restarts are available, specifically the increase from its default value by a factor of two. Trying small populations first is logical as they come along with low search costs. Therefore, an automated restart mechanism with increasing population size has been proposed and successfully employed [18].

3.3. A Common Concept

Apart from being inspired by nature, PSO and CMA-ES share important common concepts. They both are set-based and stochastic search procedures—the iteration is essentially based on a *set* of solution points, rather than on a single solution. Set-based stochastic search algorithms aim to perform well also in non-smooth or multi-modal search landscapes. More interestingly though, both algorithms are essentially based on a momentum equation of virtually identical nature. For particle $\boldsymbol{x}^t$, the update of the velocity vector $\boldsymbol{v}^t$ with discount factor $w \approx 0.72 < 1$ and stochastic "input" $\boldsymbol{x}^{t+\frac{1}{2}} - \boldsymbol{x}^t$, as developed in Equation (8), reads

$$\boldsymbol{v}^{t+1} = w \, \boldsymbol{v}^t + (\boldsymbol{x}^{t+\frac{1}{2}} - \boldsymbol{x}^t) \ , \tag{17}$$

where $\boldsymbol{x}^{t+\frac{1}{2}}$ is a perturbed average of the best positions $\boldsymbol{p}$ and $\boldsymbol{g}$. The velocity accumulates the movements of a single particle toward a disturbed average good position in the iteration sequence. This equation has a close conceptual counterpart in the Equations (11) and (12) of CMA-ES, where the notion of an *evolution path*, or *cumulation*, has been used [5]. The evolution path accumulates the movements of the population mean with a discount factor $\eta \approx n/(n+4)$. For a single-parent population, where $\boldsymbol{m}^t$ equals to the best individual of the former iteration step, the update in Equation (12) reads

$$\boldsymbol{p}^{t+1} = \eta \, \boldsymbol{p}^t + \frac{\sqrt{1 - \eta^2}}{\sigma^t} \, (\boldsymbol{m}^{t+1} - \boldsymbol{m}^t) \ . \tag{18}$$

Both equations (17) and (18) implement an exponential smoothing of movements in different time steps. In Equation (18) the choice for the discount factor corresponds to a backward time horizon of $\frac{1}{1-\eta} \approx n/4+1$ iterations (the recommended range is between $\sqrt{\frac{n}{2}}$ and n, see [5]). The value contrasts the constant choice of $\frac{1}{1-w} \approx 3.6$ in s-PSO. A second difference arises in the usage of $\boldsymbol{p}^{t+1}$ and $\boldsymbol{v}^{t+1}$. The evolution path in the CMA-ES guides new mutations only in that $\boldsymbol{p}^{t+1}$ and $-\boldsymbol{p}^{t+1}$ are equivalent. Mutation steps in both directions are equally probable. In contrast, the velocity in PSO guides in a directional way and has a substantial influence on the (expected) position of the particle in future.

4. Methods and Test Functions

4.1. Test Functions

We use the small set of well-established benchmark functions given in Table 1. All functions are tested in their original axis-parallel version (i.e. $\boldsymbol{B}$ is the identity and $\boldsymbol{y} = \boldsymbol{x}$), and in rotated versions, where $\boldsymbol{y} = \boldsymbol{B}\boldsymbol{x} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n] \, \boldsymbol{x}$. The orthogonal matrix $\boldsymbol{B}$ is chosen such that each $\boldsymbol{b}_i$ is uniformly distributed on the unit hypersphere surface [5], fixed for each run. We shall now discuss each function in turn.

Table 1: Test functions Ellipsoid, Rosenbrock, Diff-Powers and Rastrigin and target function values, where $\boldsymbol{y} := \boldsymbol{B}\boldsymbol{x}$ and $\boldsymbol{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n]$ implements an angle-preserving, linear transformation, *i.e.* $\boldsymbol{B}$ is orthogonal. The search domain is $[-20, 80]^n$ in all cases

Function	α	f_{target}
$f_{\text{elli}}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha^{\frac{i-1}{n-1}} y_i^2$	$[1, 10^{10}]$	10^{-9}
$f_{\text{Rosen}}(\boldsymbol{x}) = \sum_{i=1}^{n-1} (\alpha (y_i^2 - y_{i+1})^2 + (y_i - 1)^2)$	$[1, 10^{8}]$	10^{-9}
$f_{\text{diffpow}}(\boldsymbol{x}) = \sum_{i=1}^{n} y_i^{2+\alpha \frac{i-1}{n-1}}$	$[0, 10]$	10^{-14}
$f_{\text{Rastrigin}}(\boldsymbol{x}) = 10\,n + \sum_{i=1}^{n} (y_i^2 - 10\cos(2\pi y_i))$		10^{-9}

The ellipsoid is a convex-quadratic function. The parameter α is the condition number of the Hessian matrix which will be varied between 1 and 10^{10} in our experiments. For $\boldsymbol{B} = \boldsymbol{I}$, the Hessian matrix is diagonal. If $\alpha = 1$ the ellipsoid is the isotropic sphere function.

The Rosenbrock function is non-separable and no single rotation can be applied that renders the variables independent. The function has its global minimum at $\boldsymbol{x} = [1, 1, \ldots, 1]$ and, for large enough α and n, one local minimum close to $\boldsymbol{x} = [-1, 1, \ldots, 1]$, see also [19]. The probability to end up in the local optimum is to our experience clearly smaller than 50%. In the Rosenbrock function the parameter α tunes the width of the bent ridge that guides to the global optimum. In the classical Rosenbrock function α equals 100. For smaller α the ridge becomes wider and the function becomes less difficult to solve. We vary α between one and 10^8.

The Diff-Powers function takes the variables to different powers. The function cannot be solved by applying a constant scaling between variables. The sensitivity differences between the variables increase with decreasing distance to the optimum: the closer to the optimum, the more difficult it gets to approach it further. The parameter α, varied between 0 and 10, determines the largest exponent and was originally set to 10 [5].

The Rastrigin function is highly multi-modal. While we are mainly interested in investigating the effect of ill-conditioning, the Rastrigin function serves to find a possible trade off between the ability to effectively conduct local search on ill-conditioned functions versus the ability to effectively search globally in a highly multi-modal topography.

For $\boldsymbol{B} = \boldsymbol{I}$, all functions but the Rosenbrock function are separable. Otherwise only the ellipsoid function for $\alpha = 1$ and the Diff-Powers function for $\alpha = 0$ remain separable.

4.2. Experimental Setup

4.2.1. Algorithms and Parameters

For Particle Swarm Optimization the *Standard PSO 2006* C-code[8], "*validated by [...] James Kennedy and Maurice Clerc*" was translated into Scilab. Also for CMA-ES, Scilab-code was used. All default parameters were applied including swarm size and population size accordingly. For search space dimensions $n = 10; 20; 40$ the swarm size was $16; 18; 22$ and the population size was $10; 12; 15$, while on the Rastrigin function the swarm size for s-PSO and the population sizes for CMA-ES were varied between 10 and 1000. The boundaries for s-PSO were adjusted to the domain $[-20, 80]^n$, such that the optimum is not in the middle of the search domain. If a component of a particle happens to be outside the search domain, it is set back on the domain boundary and the corresponding velocity component is set to zero. No boundaries were applied to CMA-ES. The default termination criteria were adjusted as described below.

4.2.2. Initial Conditions

The initial swarm for s-PSO and the initial distribution mean for CMA-ES were sampled uniformly distributed in the domain $[-20, 80]^n$. The initial velocities, according to Equation (8), were sampled for each particle as half of its way to another uniformly sampled solution point (default in s-PSO). The initialization disfavors axis-parallel oriented initial velocities, but we assume that the influence on the results is marginal. The initial σ^0 for CMA-ES was 100/3 and by default $\boldsymbol{C}^0 = \boldsymbol{I}$ and $\boldsymbol{p}_\sigma^0 = \boldsymbol{p}_{\text{c}}^0 = \boldsymbol{0}$. The initialization policy were the same for rotated and non-rotated functions.

4.2.3. Termination Criteria

A run was terminated when the target function value according to Table 1 was reached or the maximum number of function evaluations 10^7 was exceeded. In order to reduce CPU-time consumption, the CMA-ES was additionally terminated when the population had converged (on the Rosenbrock and the Rastrigin function). While this is a disadvantage in principle, the probability that it has affected the outcome of any of the presented results is negligible. For s-PSO this approach proved to be infeasible, because the swarm did not regularly converge to a single point.

4.2.4. CPU Timing

The CPU time consumption of both algorithms were tested on the Rosenbrock function f_{Rosen} in dimensions 2, 3, 5, 10, 20, 40, 80 on an Intel® Core™ 2 6700 (2.66GHz) using Scilab 4.1.2 in Ubuntu i686 2.6.32-25-generic. An algorithm is restarted at least two times and until at least thirty seconds have passed. Table 2 reports CPU milliseconds per function evaluation. The larger times in lower dimensions reflect the larger proportion of initialization overhead versus actual iterations, simply because a smaller number of iterations is necessary to solve the function in smaller dimension. Leaving aside the initial overhead, s-PSO appears to be two to three times faster

[8] `http://www.particleswarm.info/Standard_PSO_2006.c`

Table 2: CPU Time per function evaluation on the Rosenbrock function in milliseconds

Alg.	2-D	3-D	5-D	10-D	20-D	40-D	80-D
PSO	3.9	1.0	0.42	0.43	0.45	0.47	0.51
CMA	13	13	5.5	4.2	1.4	0.87	1.5

than CMA-ES. Naturally, such results strongly depend on the programming language and the specific implementations. For larger dimensions we ideally expect s-PSO implementations to scale linearly with the dimension and CMA-ES implementations quadratically. This expected scaling behavior cannot be observed up to the shown dimensionality of 80.

4.2.5. Experiments

In each experiment 21 trials were conducted and for each trial on a rotated function a new basis $\boldsymbol{B}$ was chosen. The same bases were used for both s-PSO and CMA-ES.

4.2.6. Performance Measures

We consider a trial, or run, to be *successful*, if the target function value is reached before 10^7 function evaluations are exceeded. The *success rate* is the ratio of successful trials in an experiment of usually 21 trials. The so-called *success performance* measures the number of function evaluations needed to reach the target function value, taking into account successful *and unsuccessful* runs. An estimator for the success performance $\widehat{\mathrm{SP1}}$, as used in [16], analyzed in [20], and also denoted as Q-measure in [21], is computed as the average number of function evaluations for the *successful* trials, divided by the success rate. The $\widehat{\mathrm{SP1}}$ is an estimate for the expected number of function evaluations to reach the target function value (with probability one), when the algorithm is applied repeatedly, given that termination of unsuccessful runs can be accomplished such that the expected run length of unsuccessful runs equals the average run length of successful runs [20].

4.2.7. Statistical Procedures

When we state observing a difference between two results in the following, we have always asserted its statistical significance. Tests for statistical significance were conducted using either the non-parametric Mann-Whitney rank sum test, or Pearson's χ^2 test for the binomial success probabilities. For the latter the function `chisq.test` from the free software environment R was used and the p-value was simulated. Statistical significance is assumed for $p < 0.01$. In order to derive a dispersion measure for $\widehat{\mathrm{SP1}}$, bootstrapping was used [22]. The empirical bootstrap distribution was sampled 10^4 times. Significance was reasoned when the 5%-ile of one distribution was larger than the 95%-ile of the other.

5. Experimental Results

We present empirical results on the four functions from Table 1. On the Ellipsoid, Rosenbrock, and Diff-Powers functions, we vary the conditioning parameter α and measure the performance of s-PSO and CMA-ES depending on this variation.

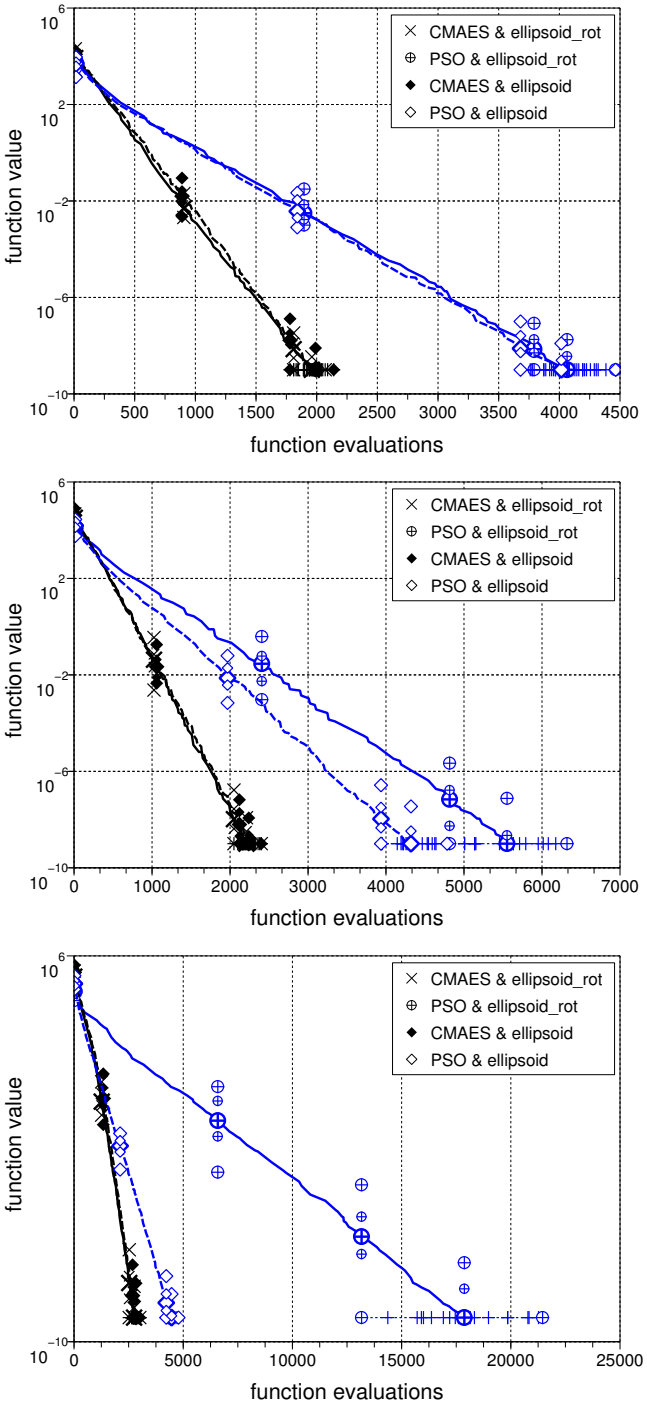

Figure 2: Ellipsoid function in 10-D: time evolution of the median function value with a function condition number α of 1 (top), 10 (middle), and 100 (bottom). Small symbols indicate the 25% and 75%-ile, large symbols indicate smallest and largest value from 21 trials. Solid lines: rotated function, dashed lines: non-rotated (separable) function. s-PSO: ⊕, ⋄; CMA-ES: ×, ⧫

5.1. Ellipsoid Function

Figure 2 depicts the evolution of the median function value of 21 runs for condition numbers 1, 10, and 100 for s-PSO and CMA-ES. In all trials the target function value was reached. In all cases we observe log-linear convergence, while the convergence rates differ depending on condition number and rotation.

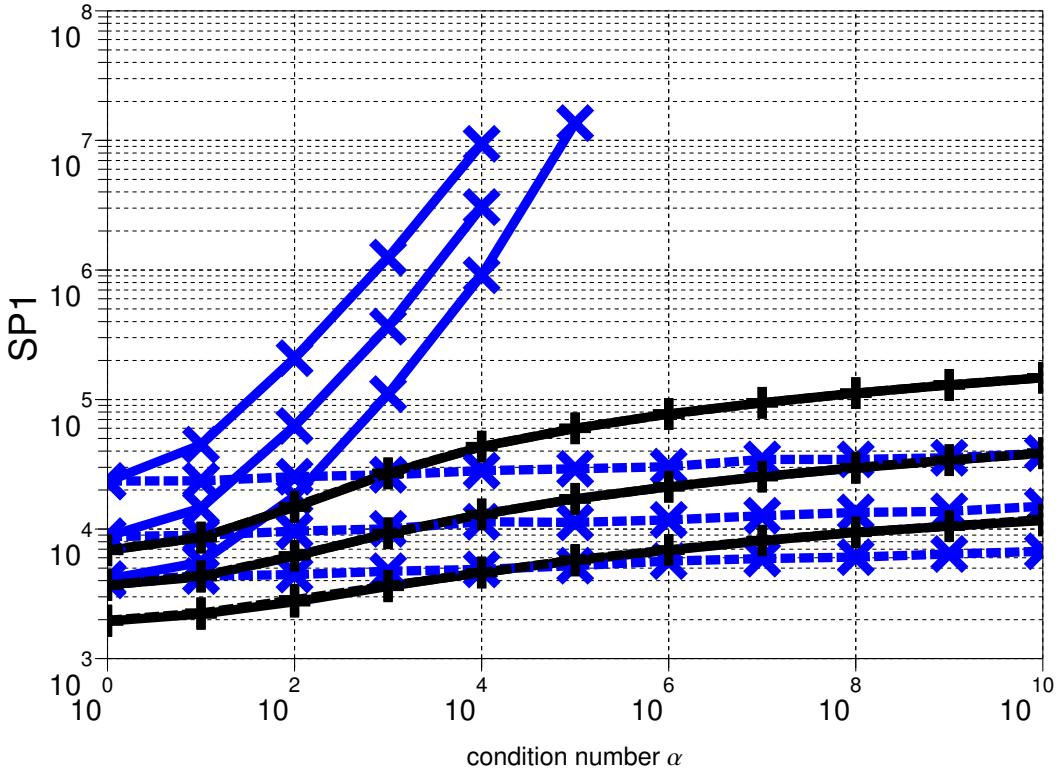

Figure 3: Ellipsoid function: $\widehat{\text{SP1}}$ (number of function evaluations) to reach $f_{\text{target}} = 10^{-9}$ on the rotated (solid) and the non-rotated (dashed) function in dimensions $10, 20, 40$ (bottom to top, respectively) over condition number α. s-PSO: $\times$, CMA-ES: $+$. Up to the condition number of 10^3 the success rates are 100%. For condition numbers larger than 10^4, s-PSO regularly exceeds the maximum number of function evaluations of 10^7 on the rotated Ellipsoid.

For condition number $\alpha = 1$, the rotated and the non-rotated function are identical and CMA-ES converges approximately twice as fast as s-PSO. With increasing condition number, the results for s-PSO on the rotated versus the non-rotated function become different. For a condition number of 100 (lowest figure), s-PSO is already about four times slower on the rotated function than in the non-rotated separable function.

Summarized performance results on the Ellipsoid function for all dimensions and condition numbers are shown in **Figure 3**, where $\widehat{\text{SP1}}$ is plotted versus the condition number α. The dependency of s-PSO on the rotation of the ellipsoid is dramatic. In the non-rotated case s-PSO works very well for all tested condition numbers and outperforms CMA-ES for large condition numbers by a factor of up to four. In the rotated case the performance degrades fast with increasing condition number. For $\alpha = 10^4$, s-PSO is already more than a hundred times slower than on the non-rotated Ellipsoid function. The rotation leads to a failure to reach the target function value before 10^7 function evaluations for condition numbers larger than 10^5 in dimension 10 and for condition numbers larger than 10^4 in dimensions 20 and 40. For condition numbers larger than 100, $\widehat{\text{SP1}}$ becomes roughly proportional to α, that means the necessary number of function evaluations increases linearly with an increasing condition number. For CMA-ES, the $\widehat{\text{SP1}}$ increases at most with $\alpha^{1/4}$, and for large α roughly with $\alpha^{0.1}$.

5.2. *Rosenbrock Function*

We investigate the effect of a varying parameter α. Table 3 shows the percentage of successful runs for all experiments conducted on the Rosenbrock function. For CMA-ES the success rate drops from 100% for $\alpha = 1$ to roughly 80% for $\alpha \geq 100$. There is no significant influence of the dimension, or of the rotation on the success rate. Unsuccessful trials of CMA-ES are those that end up in the local minimum of the Rosenbrock function.

The success rates of s-PSO are comparable with CMA-ES for small values of α, while they drop to zero for $\alpha \geq 10^5$. The rotation becomes visible in the resulting success rate with $\alpha = 10^3$ in 10- and 20-D. In 40-D, the success rate drops to zero already for $\alpha = 100$ on the rotated function.

The Rosenbrock function already exhibits dependencies between parameters in its original non-rotated version, where the condition parameter α is set to 100. In **Figure 4**, the time evolution of the median function value is shown in 10-D for the rotated and the original functions. Surprisingly, also on the Rosenbrock function, the rotation has a remarkable effect on the performance of s-PSO when the optimum needs to be approached (for function values smaller than about four), but similar results for the Rosenbrock function have been recently reported in [23]. On the rotated function, s-PSO is finally slower by a factor of about ten. The CMA-ES shows, as expected, the same behavior on non-rotated and rotated function and outperforms s-PSO by a factor of about ten and a hundred, respec-

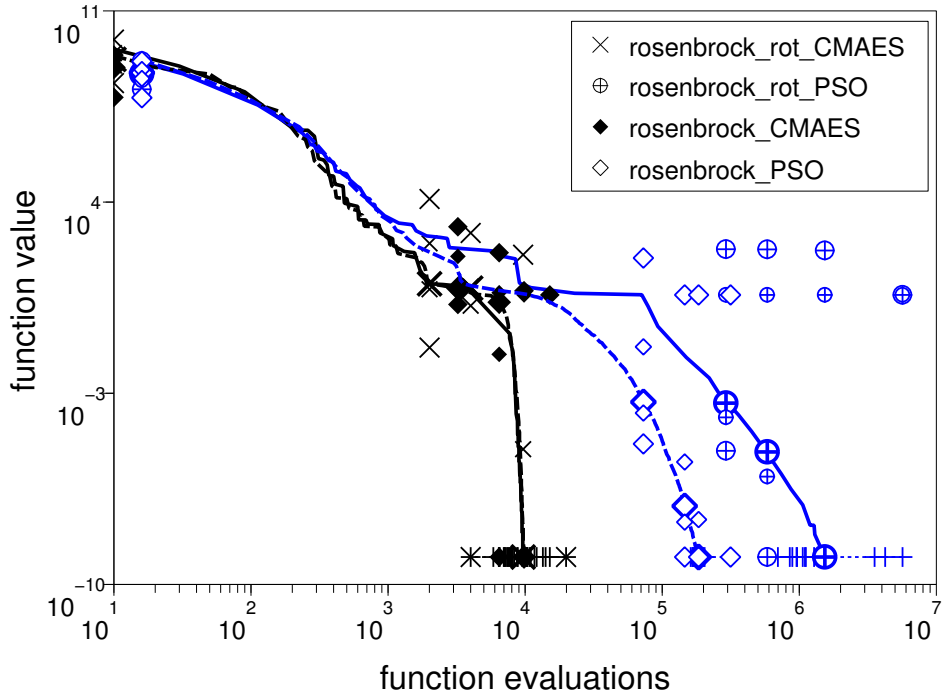

Figure 4: Rosenbrock function in 10-D with the original function condition parameter $\alpha = 100$: median function value over function evaluations, rotated case (solid lines) and non-rotated case (dashed lines).

Table 3: Rosenbrock function: percentage of successful trials (number in parentheses), out of 21. A dash (–) means that no trial reached the value 10^{-9} within 10^7 evaluations.

	dimension	α	1	10	100	300	1000
S-SPO	10		100 (21)	95 (20)	81 (17)	43 (9)	81 (17)
		rotated	100 (21)	90 (19)	71 (15)	86 (18)	–
	20		100 (21)	90 (19)	76 (16)	62 (13)	86 (18)
		rotated	100 (21)	86 (18)	62 (13)	48 (10)	–
	40		100 (21)	86 (18)	67 (14)	62 (13)	24 (5)
		rotated	100 (21)	86 (18)	–	–	–
CMA-ES	10		100 (21)	90 (19)	71 (15)	86 (18)	86 (18)
		rotated	100 (21)	100 (21)	100 (21)	90 (19)	76 (16)
	20		100 (21)	71 (15)	81 (17)	90 (19)	90 (19)
		rotated	100 (21)	76 (16)	76 (16)	86 (18)	86 (18)
	40		100 (21)	95 (20)	71 (15)	81 (17)	81 (17)
		rotated	100 (21)	100 (21)	86 (18)	81 (17)	86 (18)

	dimension	α	10^4	10^5	10^6	10^7	10^8
s-PSO	10		81 (17)	–	–	–	–
		rotated	–	–	–	–	–
	20		–	–	–	–	–
		rotated	–	–	–	–	–
	40		–	–	–	–	–
		rotated	–	–	–	–	–
CMA-ES	10		76 (16)	71 (15)	81 (17)	81 (17)	86 (18)
		rotated	81 (17)	81 (17)	81 (17)	100 (21)	86 (18)
	20		76 (16)	81 (17)	81 (17)	86 (18)	76 (16)
		rotated	86 (18)	86 (18)	90 (19)	67 (14)	76 (16)
	40		62 (13)	95 (20)	62 (13)	81 (17)	67 (14)
		rotated	90 (19)	71 (15)	81 (17)	71 (15)	76 (16)

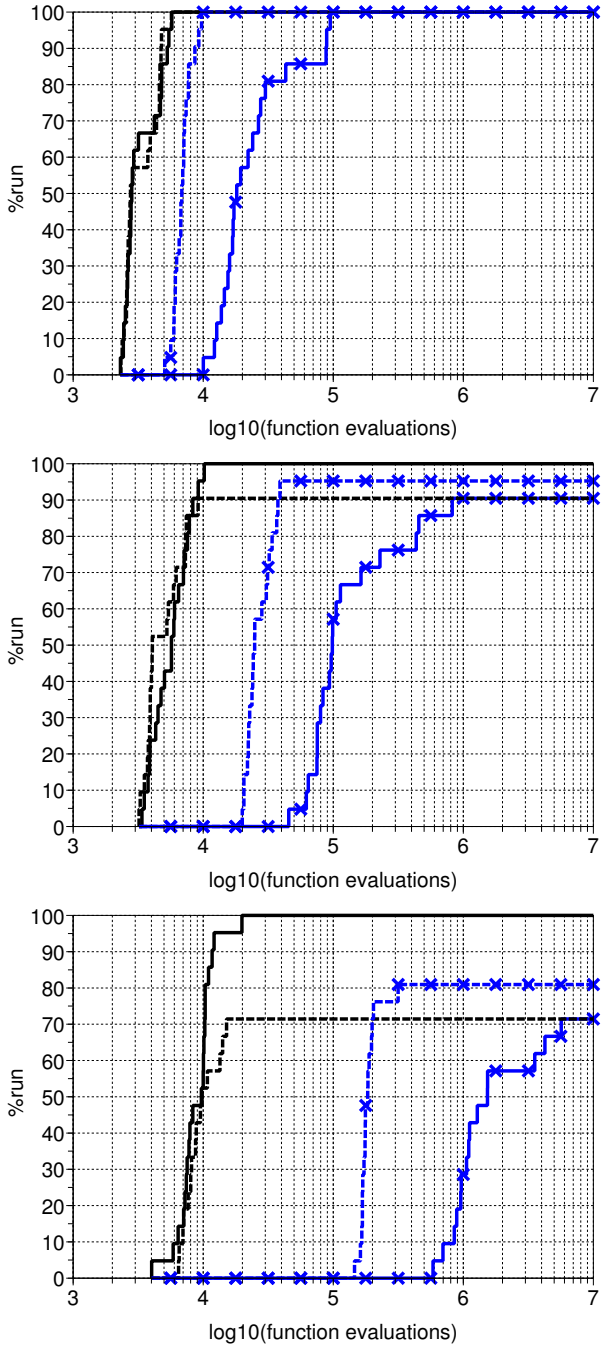

Figure 5: Rosenbrock function in 10-D: empirical cumulative distribution of the number of function evaluations to reach the target function value with a function conditioning parameter α of 1 (top), 10 (middle), and 100 (bottom). s-PSO (with small crosses) corresponds to the two lines to the right in each case. Solid lines: rotated, dashed lines: non-rotated function.

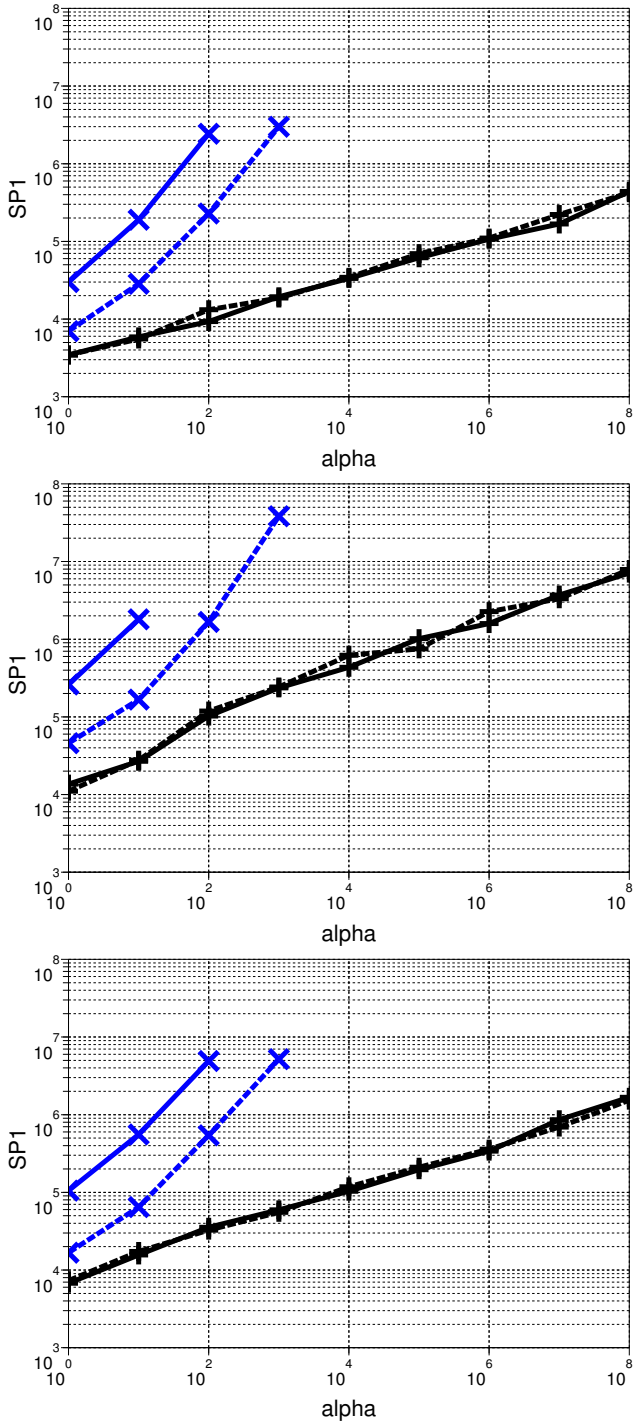

Figure 6: Rosenbrock function: $\widehat{\mathrm{SP1}}$ (number of function evaluations) versus conditioning parameter α in 10 (top), 20 (middle), and 40-D (bottom). s-PSO corresponds to the two upper lines in each case. Solid lines: rotated, dashed lines: non-rotated.

tively.

The empirical distribution of the number of function evaluations to reach the target function value is depicted in **Figure 5** for $\alpha = 1; 10; 100$ in 10-D. Even for $\alpha = 1$ the rotation significantly compromises the performance of s-PSO and the effect becomes slightly more pronounced with increasing α. In all cases the necessary number of evaluations increases with increasing α as the distribution graphs move to the right. For $\alpha = 100$, we can conjecture that some runs of s-PSO might reach the target value slightly after the maximum number of function evaluation is exceeded.

Figure 6 shows all $\widehat{\mathrm{SP1}}$ measures on the Rosenbrock function, where each line in Fig. 5 collapses to a point in Fig. 6, top, $\alpha \leq 100$. The rotation leads roughly to an increase of

$\widehat{\text{SP1}}$ by a factor of five to ten independent of α or the search space dimensionality. The sensitivity to the condition parameter α is quite pronounced and independent of the rotation: the number of function evaluations increases roughly linearly with α. From these graphs we can conjecture that the drop of the success probability for large α is most likely induced by the termination criterion of 10^7 function evaluations and does not indicate a principle failure of s-PSO.[9] Again, CMA-ES behaves empirically rotationally invariant, its $\widehat{\text{SP1}}$-performance scales between $\alpha^{1/4}$ and $\alpha^{1/3}$ and it clearly outperforms s-PSO on the Rosenbrock function.

5.3. Diff-Powers Function

Experiments on the Diff-Powers function are shown in **Figure 7** for $\alpha = 0, 2, 10$. For $\alpha = 0$, the isotropic, quadratic sphere function is recovered, as it was the case for the ellipsoid function.

The effect of α on the performance is quite moderate on the non-rotated function for both algorithms, which also perform similar. An algorithm that cannot adapt the scaling would show a dramatic drop in performance with increasing α, as reported in [5]. This can also be observed for s-PSO in the rotated case. The performance is entirely different from the non-rotated case. For $\alpha = 2$, s-PSO needs 500 times longer than in the non-rotated case to reach the target function value. By extrapolating the graph for $\alpha = 10$ we estimate the scaling of the number of function evaluations as roughly $\propto 10^{\alpha}$, for the given target function value.

Figure 8 depicts $\widehat{\text{SP1}}$ and reveals that only for $\alpha \leq 2$ s-PSO reaches the target function value on the rotated function, while CMA-ES slows down by a factor of up to four for α approaching ten. In the non-rotated case, on the other hand, s-PSO becomes even slightly faster with increasing α. Can we explain this behavior? The Diff-Powers function is generally more difficult to solve with increasing α, because the different sensitivities of the parameters become more pronounced and the topography becomes less spherical. On the contrary, in order to reach the same target function value, when α increases, the parameters with large an exponent need to be located with lesser precision. The search space volume for which the function value is smaller than the target function value increases. For this reason, s-PSO becomes slightly faster on the non-rotated function and for the same reason also the target function value was chosen considerably smaller than 10^{-9}.

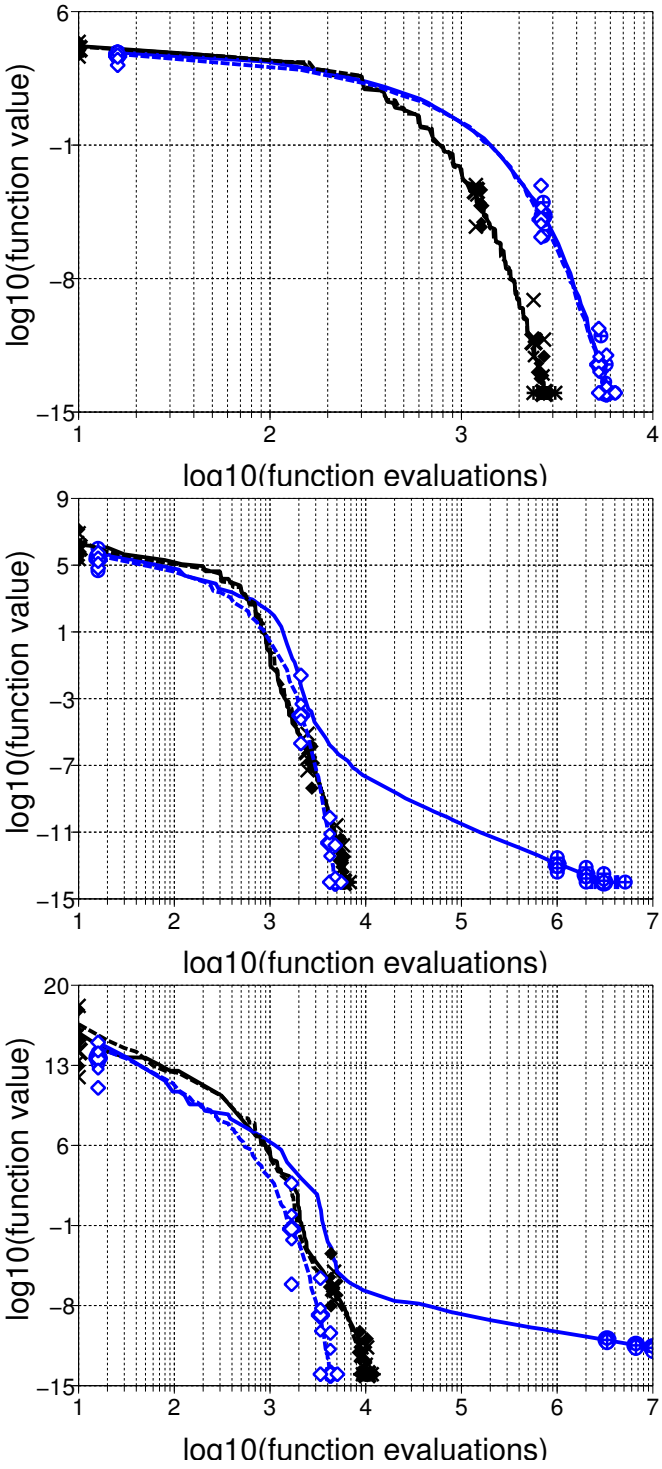

Figure 7: Diff-Powers function in 10-D: time evolution of the median function value, rotated and non-rotated case, with parameter α equals to zero (top), two (middle) and ten (bottom). Small symbols indicate the 25%- and 75%-ile, large symbols indicate smallest and largest value from 21 trials. s-PSO: $\oplus, \diamond$; CMA-ES: $\times, \blacklozenge$

5.4. Rastrigin Function

The Rastrigin function is a highly multi-modal test function and not easy to solve. Our tests on the Rastrigin function serve to check whether on a multimodal function a similar dependency on rotation can be found and whether a trade off between local and global search performance can be observed. Here, the function is kept globally isotropic (no condition parameter).

On multi-modal functions the swarm and population size become a decisive factor. Therefore different sizes between 10

[9] From a methodological view point it is interesting to notice that this interpretation is only possible by using a quantitative performance measure that can be extrapolated and that it becomes apparent in an appropriate visualization. Reporting success rates alone appears to be misleading and therefore insufficient.

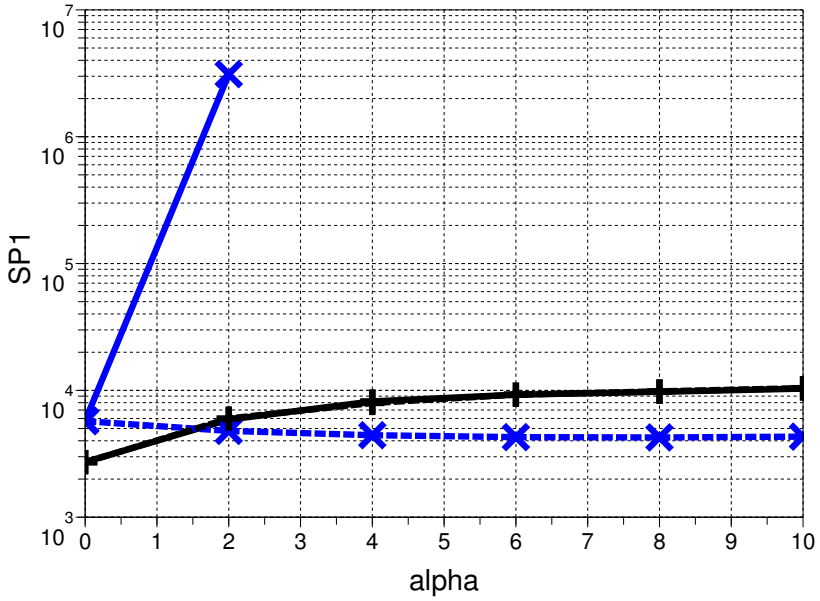

Figure 8: Diff-Powers function in 10-D: $\widehat{\mathrm{SP1}}$ (number of function evaluations) versus parameter α. s-PSO: $\times$, CMA-ES: +; solid lines: rotated, dashed lines: non-rotated function. For CMA-ES both graphs are virtually identical. Missing points indicate that 10^7 function evaluations were exceeded in all trials. The axis parallel function becomes with increasing α even more easy to solve for s-PSO.

and 1000 were investigated. **Figure 9** shows the time evolution of the function value from all 21 runs on the non-rotated (left) and rotated Rastrigin function for swarm/population sizes of 30, 100, 300, and 1000. For CMA-ES, only results on the rotated function are shown (right column). Results on the non-rotated function are congruent. **Table 4** tabulates all corresponding success rates. The CMA-ES reaches a success probability larger than 50% for a population size of 300. On the non-rotated function, s-PSO needs a swarm size of 100 for a comparable success probability. The variation between different trials is much more pronounced for s-PSO. In contrast to CMA-ES, in s-PSO the particles retain diversity for a long time and can eventually find the global optimum even after more than one million function evaluations.

Also on the Rastrigin function the results of s-PSO are entirely different for the rotated versus the non-rotated case. The separable Rastrigin function can be solved reliably, if the swarm size is chosen larger than 100. In contrast, the rotated Rastrigin function was solved in only one trial (with swarm size 100). In 160 additional trials in this set-up, 80 with the same basis $\boldsymbol{B}$, 80 with a new basis (see Table 1) another single success was observed and we conclude that the success rate is roughly 1%. This conclusion is in agreement with all presented data, but beyond the sensitivity of our experimental set-up.

Figure 10 shows empirical cumulative distribution functions of the number of function evaluations to reach the target function value on the left part, and of the best function value achieved at maximum number of evaluations 10^7 on the right from all experiments with swarm/population sizes between 30 and 1000. The value of the graph at the transition between left and right reflects the success rate. This presentation allows for a better visual comparison than in Figure 9. For the smallest swarm/population size all results are fairly similar. For swarm/population size 100 on the separable function s-PSO has a clearly improved performance. For swarm/population size 300 and 1000 CMA-ES shows the better result.

These result have been even more condensed by displaying $\widehat{\mathrm{SP1}}$ in **Figure 11**. The variance of $\widehat{\mathrm{SP1}}$ for a small swarm/population size is large as the outcome depends on a small number of successful runs. Yet the graphs are rather flat, in that the $\widehat{\mathrm{SP1}}$ measure is comparatively insensitive to the swarm/population size, as long as the function was solved at all. This result can partly be attributed to the small number of trials (for swarm/population size of 30 only one out of 21 trials was successful respectively) and will not hold for even smaller or larger swarm/population sizes. Invariably s-PSO is roughly ten times slower than CMA-ES while the statistical significance of this difference is only asserted for $S = \lambda \geq 300$.

The strong dependency of s-PSO on the rotation of the Rastrigin function comes as a surprise to us. Seemingly, s-PSO can exploit the axis-parallel alignment of local optima, either by exploiting the given axis-parallel bounds or due to its variation mechanism. The former explanations seem rather implausible, in particular as the boundaries have been chosen "loose" such that any solution close to the boundaries is of very low quality.

On the (non-rotated) Rastrigin function it is generally not sufficient to align the swarm along *one* coordinate axis. For successful basin hopping in high dimensions large steps must be taken along *different* coordinate directions, in order to locate the global optimum based on the function's separability. In or-

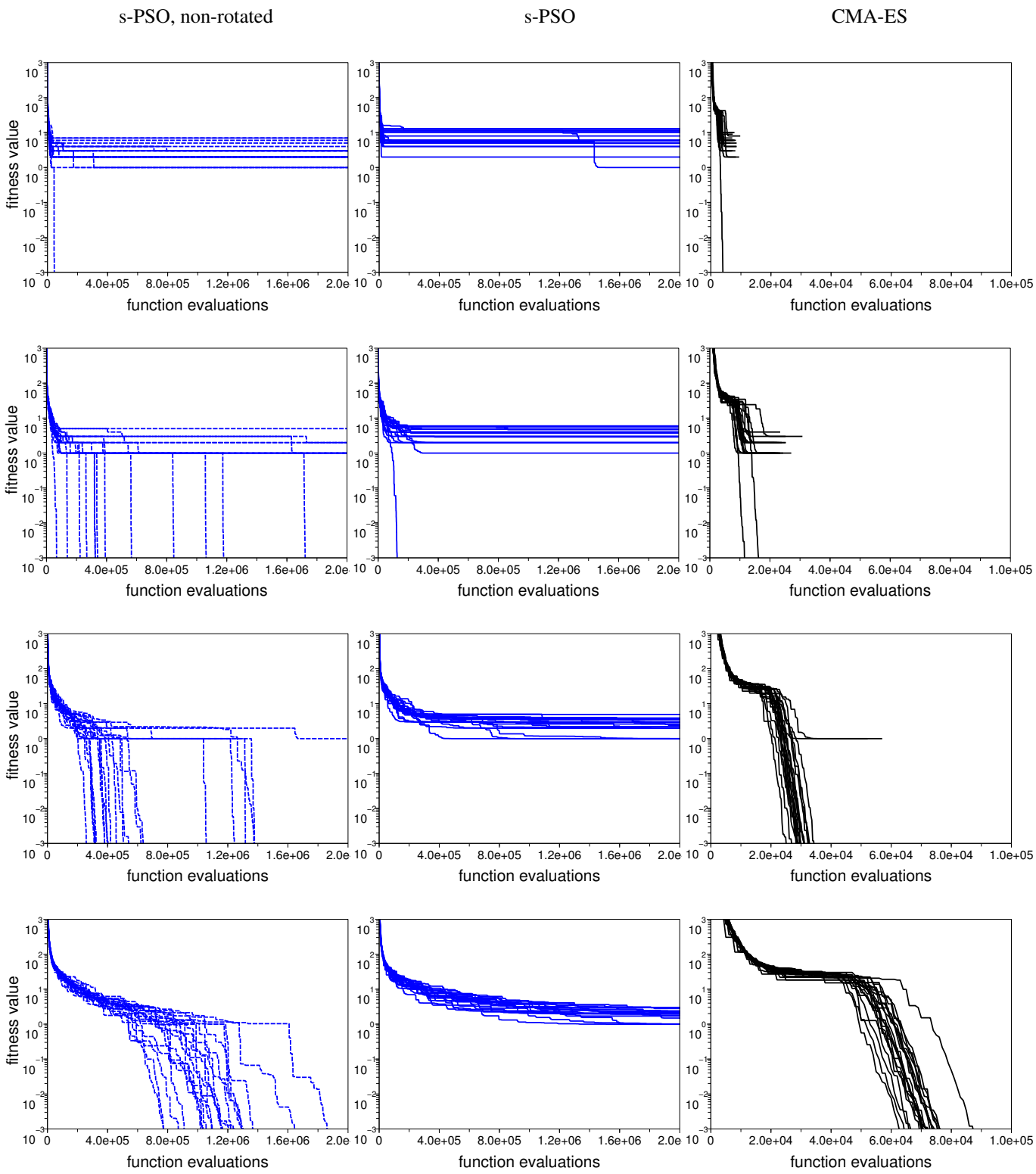

Figure 9: Rastrigin function in 10-D: time evolution of the objective function value of all 21 runs. Left column: s-PSO on the non-rotated Rastrigin function until up to 2×10^6 function evaluations. Middle column: s-PSO on the rotated Rastrigin function until up to 2×10^6 function evaluations. Right column: CMA-ES on the rotated Rastrigin function until up to 10^5 function evaluations. Swarm/population size of 30, 100, 300, 1000 from top to bottom. The CMA-ES was terminated before the maximum number of function evaluations was reached also in the unsuccessful trials.

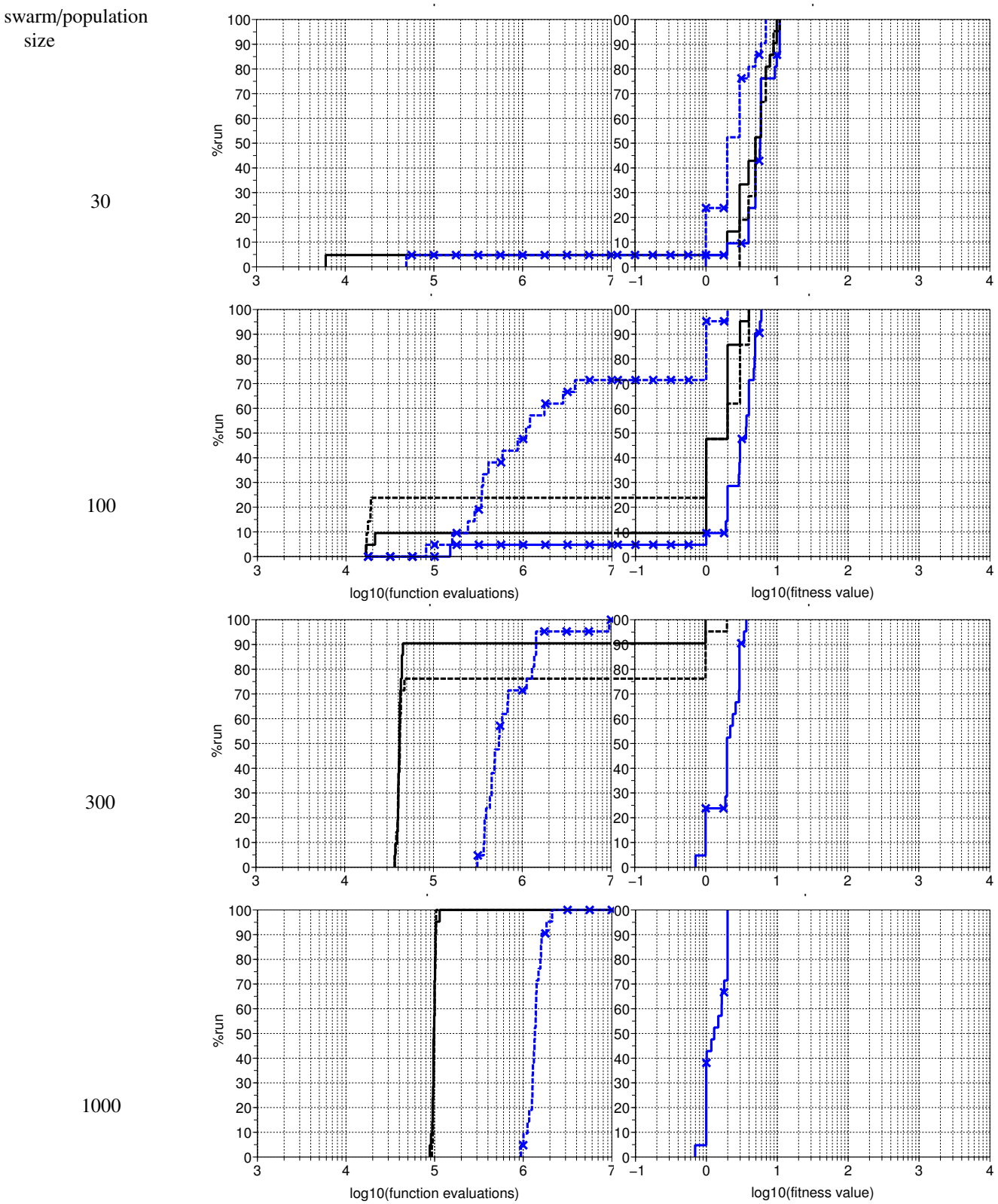

Figure 10: Rastrigin function in 10-D: empirical cumulative distribution (ECDF) of the number of function evaluations to reach the target function value 10^{-9} (left column) and ECDF of the best function value after the maximum number of function evaluations is exceeded (right column). Swarm/population size of 30, 100, 300, 1000 from top to bottom. s-PSO with small crosses, CMA-ES without; solid lines: rotated, dashed lines: non-rotated function.

Table 4: Rastrigin function in 10-D: percentage of successful trials (number in parentheses), out of 21, for different swarm/population sizes

S, λ	10&16	30	100	300	1000
s-PSO	–	5% (1)	71% (15)	100% (21)	100% (21)
rotated	–	–	5% (1)	–	–
CMA-ES	–	–	24% (5)	76% (16)	100% (21)
rotated	–	5% (1)	10% (2)	90% (19)	100% (21)

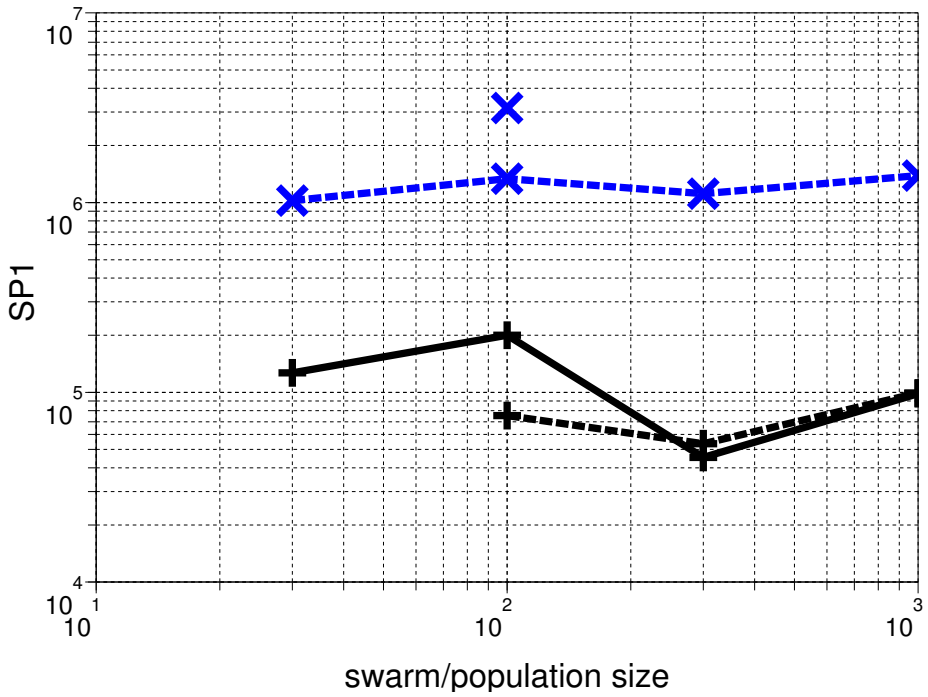

Figure 11: Rastrigin function in 10-D: $\widehat{\text{SP1}}$ (number of function evaluations) versus swarm size/population size. s-PSO: × (above lines), CMA-ES: + (below lines); solid lines: rotated, dashed lines: non-rotated function. For missing points no trial reached the target function value. $\widehat{\text{SP1}}$ shows comparatively flat graphs, because the number of function evaluations to reach the target value in successful trials is negatively correlated with the success rate.

der to make large steps preferably in a coordinate direction the three vectors $\boldsymbol{x}^t$, $\boldsymbol{x}^{t-1}$, and $\frac{p+g}{2}$ must be aligned in all but a few components, say two or three. This seems to happen systematically. Additional experiments in 20-D with swarm size 2000 also gave 100% success rate and make the moderate dimension as explanation factor implausible.

6. Discussion

Rotational Invariance and Search Space Boundaries. In our experimental design the search space boundaries are defined in the given coordinate system also for rotated functions. This means, the initial distribution mean for CMA-ES and the initial swarm and velocities for s-PSO are always chosen "non-rotated". For a "complete" change of coordinate system also the boundaries need to be rotated. On the one hand, the chosen experimental setup is closer to practice, where the search space boundaries are (trivially) defined in the given coordinate system. On the other hand, the setup might have an influence on the observed "rotational invariance" and a truly rotationally invariant algorithm might perform differently in the rotated and non-rotated cases. For the rotationally invariant CMA-ES this effect was not visible justifying our practically oriented approach.

Parameters. We believe that a standard parameter settings that works well over a wide range of objective functions is an essential feature for the applicability of any search algorithm. For this reason we have omitted parameter tuning and used default settings. In practice, when the objective function is costly, tuning parameters is prohibitive. For tuning only the population/swarm size on the Rastrigin function a comparatively effective method is available [18]. On the other hand, accompanying experiments revealed no evidence that the obtained results are highly sensitive to any parameter tuning or minor algorithmic tweak, in that much better results could have been achieved with a simple adjustment. Worse result can easily be provoked by small and/or simple modifications.

No Free Lunch. On the non-separable functions of our small test function set, s-PSO is consistently outperformed by the CMA-ES. Our benchmark functions were specifically selected

to test the behavior on non-separable, ill-conditioned problems, where dependencies between the design parameters play a decisive role. One might argue that this implies s-PSO must be better on other benchmark functions or on real world problems as the *no free lunch* (NFL) theorem seems to implicate [24, 25]. This would render experimental results fairly meaningless for practical implications and would indeed be true, if the *necessary* conditions for NFL would hold, namely, if the set of all considered functions were closed under permutation [26]. Fortunately, we do not have much reason to believe that this condition holds on any interesting set of test and/or real world problems (including *all* interesting or all ever considered problems) [27, 28]. Additionally, in continuous domain, NFL seems not to be available at all [29]. After all, NFL theorems can hardly have a grave impact on the relevance of such empirical findings.

7. Summary and Conclusion

We summarize the findings of this article, where we compared *standard Particle Swarm Optimization 2006* (PSO) and *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES). An empirical investigation was conducted on four *parameterized* test functions with condition numbers of up to 10^{10}, and search space dimensionality between 10 and 40. All parameters were held constant, besides the swarm size or population size, respectively, on the Rastrigin function. Results were mainly expressed as number of function evaluations needed to reach a target function value (search costs or running time) and therefore yield a quantitative assessment.

Ill-Conditioned, Separable Functions. PSO performs very well on ill-conditioned, separable functions, where the design variables of the objective function are independent. The search costs are almost independent of the condition of the function and correspond well to an invariance under the scaling of variables (diagonal invariance). PSO outperforms the rotationally invariant CMA-ES by a factor of about three. The latter exhibits search costs which scale roughly with $\alpha^{1/4}$, where α corresponds to a condition number of the function.

Coordinate System Rotation. The performance of CMA-ES is unaffected by coordinate system rotations (despite the coordinate system dependent setting of the initial distribution mean), which corresponds to its rotational invariance property. The performance of PSO on even moderately ill-conditioned separable functions declines remarkably with coordinate system rotations, where the design variables of the objective function become dependent. Also on the per-se non-separable Rosenbrock function, a coordinate system rotation leads to a decline in performance by a factor of about ten, nearly independently of search space dimension and conditioning parameter.

Ill-Conditioning, Non-Separable Functions. On non-separable functions the search costs of PSO increase roughly *linearly with the condition number* α of the problem, for condition numbers larger than 100. This holds true on quadratic and non-quadratic problems and is independent of the problem dimension. Our data do *not* indicate that PSO converges prematurely and fails to solve ill-conditioned, non-separable functions. However, in comparison, the CMA-ES achieves slightly better performance on well conditioned problems and search costs scale roughly between $\alpha^{1/10}$ and $\alpha^{1/3}$ with condition number α. Consequently, CMA-ES outperforms PSO roughly by a factor of $\alpha^{3/4}$. For still a moderate condition number, say 10^4, this amounts to a factor of a thousand.

Multi-Modal Functions. On the multi-modal 10-D Rastrigin function, PSO is able to locate the optimum with a large swarm size reliably in the separable case, where it is approximately ten times slower than the CMA-ES with a slightly larger population size. In the rotated case, the success probability for PSO drops to roughly 1% while the CMA-ES performs invariant under rotations. No trade-off between performance on unimodal versus multi-modal functions can therefore be reported.

A Common Concept. A close parallel of the *velocity* update in PSO and the update of an *evolution path* in ESs is elaborated. While the updates are virtually identical in their concept and formulation, two aspects are different. The velocities direct future headings of particles, while the evolution path does not affect the mean displacements in search space per se. The backward time constant for the evolution path increases proportionally with the search space dimension while the time constant for the velocity is independent of the search space dimension. We conjecture a link between these two differences.

Invariance properties. We believe invariance is an important aspect in continuous domain search. We have reviewed the most important invariances. Invariances induce equivalence classes of objective functions and consequently guarantee the generalization of performance results within each class, thereby considerably strengthening their relevance. Invariance generalizes any result and is not a priori associated with sound performance. The CMA-ES and PSO are invariant under order-preserving transformations of the objective function value. Given respective initial conditions, CMA-ES and PSO are invariant under a scaling of the search space and a scaling of single variables. Only CMA-ES is invariant under rotations of the search space and under general linear search space transformations.

Implications. We believe that ill-conditioning is a prevalent property of difficult real-world problems and that there is no trivial solution to searching *non-separable*, ill-conditioned problems efficiently. Achieving rotational invariance alone is not sufficient as isotropic algorithms necessarily perform poorly on ill-conditioned problems. In order to solve a non-separable ill-conditioned problem, first, a non axis-parallel narrow valley must be well represented in the search algorithm. Second, collapsing into low-dimensional subspaces must be prevented. Rotational invariance then generalizes experimental results to any coordinate system rotation. For PSO, the application of *adaptive encoding* [30] could be a promising measure to achieve this task.

Acknowledgements

This work was partly funded by the *Optimisation Multi-Disciplinaire* projects (OMD – ANR/RNTL2005/OMD and OMD^2 – ANR-08-COSI-007-12) on the one hand, and by the Microsoft Research - INRIA joint laboratory (`http://www.msr-inria.inria.fr/`) on the other hand.

References

[1] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Neural Networks, 1995. Proceedings., IEEE International Conference on, Vol. 4, 1995, pp. 1942–1948.

[2] Y. Shi, R. Eberhart, Modified particle swarm optimizer, in: The 1998 IEEE International Conference on Evolutionary Computation, ICEC'98, 1998, pp. 69–73.

[3] Y. Shi, R. Eberhart, Empirical study of particle swarm optimization, in: Proceedings of the 1999 Congress on Evolutionary Computation, Vol. 3, Piscataway, NJ, IEEE Service Center, 1999, pp. 1948–1950.

[4] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, Evolutionary Computation, IEEE Transactions on 6 (1) (2002) 58–73.

[5] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evolutionary Computation 9 (2) (2001) 159–195.

[6] N. Hansen, The CMA evolution strategy: a comparing review, in: J. Lozano, P. Larranaga, I. Inza, E. Bengoetxea (Eds.), Towards a new evolutionary computation. Advances on estimation of distribution algorithms, Springer, 2006, pp. 75–102.

[7] N. Hansen, A. Ostermeier, Adapting arbitrary normal mutation distributions in evolutionstrategies: the covariance matrix adaptation, in: Proceedings of the IEEE Congress on Evolutionary Computation, 1996, pp. 312–317.

[8] S. Gelly, S. Ruette, O. Teytaud, Comparison-based Algorithms are Robust and Randomized Algorithms are Anytime, Evolutionary Computation Journal 15 (4) (2007) 411–434.

[9] V. Miranda, Evolutionary algorithms with particle swarm movements, in: Intelligent Systems Application to Power Systems, 2005. Proceedings of the 13th International Conference on, 2005, pp. 6–21.

[10] D. Wilke, S. Kok, A. Groenwold, Comparison of linear and classical velocity update rules in particle swarm optimization: Notes on scale and frame invariance, Int. J. Numer. Meth. Engng 70 (2007) 985–1008.

[11] S. Kok, D. Wilke, A. Groenwold, Recent developments of the particle swarm optimization algorithm, in: Proc of International Conference on Computational Intelligence, Vol. 2005, 2005, pp. 392–397.

[12] D. Wilke, S. Kok, A. Groenwold, Comparison of linear and classical velocity update rules in particle swarm optimization: Notes on diversity, Int. J. Numer. Meth. Engng 70 (2007) 962–984.

[13] I. Rechenberg, Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution, Frommann-Holzboog, 1973.

[14] H.-P. Schwefel, Evolution and Optimum Seeking, Sixth-Generation Computer Technology Series, John Wiley & Sons, 1995.

[15] H.-G. Beyer, H.-P. Schwefel, Evolution strategies: A comprehensive introduction, Natural Computing 1 (1) (2002) 3–52.

[16] N. Hansen, S. Kern, Evaluating the CMA evolution strategy on multimodal test functions, in: X. Yao, et al. (Eds.), Parallel Problem Solving from Nature - PPSN VIII, LNCS 3242, Springer, 2004, pp. 282–291.

[17] H.-G. Beyer, Evolution strategies, Scholarpedia 2 (8) (2007) 1965.

[18] A. Auger, N. Hansen, A restart CMA evolution strategy with increasing population size, in: Proceedings of the IEEE Congress on Evolutionary Computation, 2005, pp. 1777–1784.

[19] Y.-W. Shang, Y.-H. Qiu, A note on the extended Rosenbrock function, Evol. Comput. 14 (1) (2006) 119–126.

[20] A. Auger, N. Hansen, Performance evaluation of an advanced local search evolutionary algorithm, in: Proceedings of the IEEE Congress on Evolutionary Computation, 2005, pp. 1769–1776.

[21] V. Feoktistov, Differential Evolution: In Search of Solutions, Optimization and Its Applications, Springer-Verlag New York, Inc. Secaucus, NJ, USA, 2006.

[22] B. Efron, R. Tibshirani, An Introduction to the Bootstrap, Chapman & Hall/CRC, 1993.

[23] R. Ros, N. Hansen, A simple modification in CMA-ES achieving linear time and space complexity, in: G. Rudolph et al. (Ed.), Parallel Problem Solving from Nature (PPSN'08), LNCS, 2008, pp. 296–305.
URL `http://hal.inria.fr/inria-00287367/en/`

[24] N. Radcliffe, P. Surry, Fundamental Limitations on Search Algorithms: Evolutionary Computing in Perspective, Lecture Notes in Computer Science, Springer Verlag, New York, NY 1000 (1995) 275–291.

[25] D. Wolpert, W. Macready, No free lunch theorems for optimization, Evolutionary Computation, IEEE Transactions on 1 (1) (1997) 67–82.

[26] C. Schumacher, M. Vose, L. Whitley, The no free lunch and problem description length, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), Morgan Kaufmann, 2001, pp. 565–570.

[27] C. Igel, M. Toussaint, A No-Free-Lunch Theorem for Non-Uniform Distributions of Target Functions, Journal of Mathematical Modelling and Algorithms 3 (4) (2004) 313–322.

[28] C. Igel, M. Toussaint, On classes of functions for which No Free Lunch results hold, Information processing letters 86 (6) (2003) 317–321.

[29] A. Auger, O. Teytaud, Continuous lunches are free!, in: Proceedings of the 9th annual conference on Genetic and evolutionary computation, ACM Press New York, NY, USA, 2007, pp. 916–922.

[30] N. Hansen, Adaptive encoding: How to render search coordinate system invariant, in: G. Rudolph et al. (Ed.), Parallel Problem Solving from Nature (PPSN'08), LNCS, 2008, pp. 205–214.
URL `http://hal.inria.fr/inria-00287351/en/`