Experimental Analysis of Bound Handling Techniques in Particle Swarm Optimization

Article *in* IEEE Transactions on Evolutionary Computation · April 2013

DOI: 10.1109/TEVC.2012.2189404

3 authors, including:

Juergen Branke
The University of Warwick
206 PUBLICATIONS **10,764** CITATIONS

SEE PROFILE

Sanaz Mostaghim
Otto-von-Guericke-Universität Magdeburg
159 PUBLICATIONS **2,708** CITATIONS

SEE PROFILE

Experimental Analysis of Bound Handling Techniques in Particle Swarm Optimization

Sabine Helwig*, Jürgen Branke†, and Sanaz Mostaghim‡
*Department of Computer Science, University of Erlangen-Nuremberg, Germany
sabine.helwig@informatik.uni-erlangen.de
†Warwick Business School, University of Warwick, United Kingdom
juergen.branke@wbs.ac.uk
†Institute AIFB, University of Karlsruhe, Germany
smo@aifb.uni-karlsruhe.de

***Abstract*—Many practical optimization problems are constrained, and have a bounded search space. In this paper, we propose and compare a wide variety of bound handling techniques for particle swarm optimization. By examining their performance on flat landscapes, we show that many bound handling techniques introduce a significant search bias. Furthermore, we compare the performance of many bound handling techniques on a variety of test problems, demonstrating that the bound handling technique can have a huge impact on the algorithm performance, and that the method recently proposed as standard is generally not performing well.**

I. Introduction

Particle swarm optimization (PSO) is a versatile population-based optimization technique. Basically, particles "fly" across the fitness landscape, while a particle's movement is influenced by its attraction to the neighborhood best (the best solution found by members of the particle's social network), and its personal best (the best solution the particle has found so far). PSO has been shown to perform well for many different problem domains, see, e.g., [1], [2], [3].

An aspect that has so far attracted little attention is the issue of bound handling in PSO. Many practical optimization problems have constraints, ranging from complex non-linear constraints to simple constraints on the range of values a variable is allowed to take on. The latter are usually called "bounds", and our paper deals particularly with bound handling, although many of the ideas probably also apply to general constraint handling.

If the problem includes bounds, it is important to properly deal with particles moving out of the feasible space. This is particularly important if, as is often the case, the optimal solutions lie close to the boundary of the feasibly space. Also, it has been shown that as the number of decision variables increases, the probability of particles moving out of the feasible area grows dramatically [4], [24], thus a proper bound handling technique becomes indispensable.

Bound handling and constraints have long been researched in evolutionary computation, and popular methods include penalizing infeasible individuals, repairing infeasible individuals, or consider bound violation as an additional objective. For a survey on constraint handling techniques in evolutionary computation see, e.g., [5], [6]. Basically, all the proposed approaches modify either the fitness or the location of solutions that lie outside the feasible region. For PSO, also the velocity of a particle needs to be taken into account. A particle's velocity introduces some sort of momentum into particle movements, and, if unaltered, will quite likely cause a particle that crossed the boundary to remain in the infeasible area.

In this paper, we have a closer look at the bound handling techniques suggested for PSO in the literature, and also propose a new one. We empirically examine a large set of bound handling techniques on a flat landscape as well as a large set of benchmark problems. The results give insights into the advantages and disadvantages of the different methods. In particular, they show that the current "standard" way of handling bounds is generally inferior.

The outline of this paper is as follows. We start with a brief introduction to PSO in Section II. In Section III, we discuss and categorize previous proposals on bound handling in PSO. The methods we will consider in this study are summarized in Section IV. In Section VI, we expose the bias some of these methods impose on search by analyzing their behavior on a flat landscape. Their performance on a variety of benchmark problems is analyzed in Section VII. The paper concludes with a summary and outlook in Section VIII.

II. Particle swarm optimization

As has already been explained in the introduction, in PSO particles move through the search space driven by attraction to their personal best found solution so far, and the best found solution in the neighborhood.

A particle i is defined by its position $\vec{x}_i$, the position of its personal best solution found so far, $\vec{p}_i$, and its velocity $\vec{v}_i$. Furthermore, each particle knows the best solution found so far by any of its "neighbors", usually called $\vec{g}_i$. Different particle topologies are explored in [7], [8], but the standard neighborhoods are *global* (all particles know the position of the globally best solution found so far), *grid* (particles are connected in a toroidal grid structure with each particle being neighbor to its four adjacent grid points), and *ring* (particles are connected via a ring structure with each particle having exactly two neighbors).

In the beginning, particle positions and velocities are generated randomly. The algorithm then proceeds iteratively, updating first all velocities and then all particle positions as follows:

$$\vec{v}_i = \chi[\vec{v}_i + c_1\vec{\epsilon_1} \cdot (\vec{p}_g - \vec{x}_i) + c_2\vec{\epsilon_2} \cdot (\vec{p}_i - \vec{x}_i)] \quad (1)$$

$$\vec{x}_i = \vec{x}_i + \vec{v}_i. \quad (2)$$

The constriction factor $\chi < 1$ acts like friction, slowing the particles, so that finer exploration is achieved. The inclusion of the previous generation's velocity in the calculation of the new velocity introduces some sort of momentum into the particle's movement. c_1 and c_2 control the relative attraction to the global best and personal best found solutions, respectively. Finally, $\vec{\epsilon_1}$ and $\vec{\epsilon_2}$ are vectors of random variables drawn with uniform probability from $[0, 1]$, and redrawn in every iteration for each particle.

Clerc and Kennedy [9] have proposed to set $c_{1,2}$ = 2.05 and $\chi = 0.729843788$, a setting that has been widely adopted by many researchers.

III. Related Work

In this section, we survey various methods for boundary handling. They are categorized into strategies just changing the personal best update, strategies that set the infeasible particle onto the boundary, strategies that prevent the particle to reach the boundary, and others.

A. Change the personal/global best update

The simplest and most straightforward method for handling the particles which cross the boundary of the feasible region is to leave their velocity and position unaltered and simply prevent to select them as personal or global best particles. Bratton and Kennedy propose this as standard methodology [10]. An advantage of this method is that the fitness evaluation of the infeasible individuals may be skipped, as these particles are simply assigned a very high (infinite) fitness value. The hope is that such particles will eventually be drawn back into the space by the influence of their personal best and neighborhood's best. We refer to this method as **infinity**.

Toscano Pulido and Coello Coello [11] suggest an approach that selects the global best particles depending on their closeness to the borders. The closer to the border, the less is the probability to select this particle as a global best particle.

B. Reposition infeasible particles

The second group of boundary handling techniques repositions the infeasible particle and places it back into the feasible area.

In the **Nearest** method, the particle is simply set onto the boundary if it exceeds the boundary, in each dimension i:

$$\text{If } X_{i,t+1} > X_{i,max}, \text{ then } X'_{i,t+1} = X_{i,max} \quad (3)$$

$$\text{if } X_{i,t+1} < X_{i,min}, \text{ then } X'_{i,t+1} = X_{i,min} \quad (4)$$

where $X_{i,max}$ and $X_{i,min}$ are the upper and lower bound for dimension i, respectively.

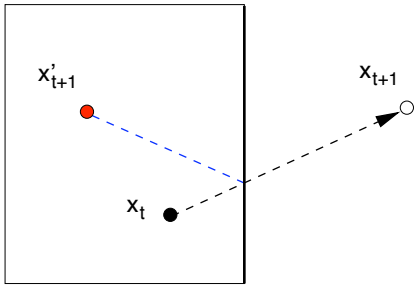


Fig. 1. The Reflect Method. The particle violates the vertical boundary and the new position is reflected into the search space.

Instead of setting the particle onto the boundary, the **Pbest** method proposed in [12] repositions the particle onto their personal best position. The **reflect** method [18] reflects the position in the search space for a particle violating a border. Figure 1 illustrates the main idea in two dimensional search space where the infeasible solution is reflected in the feasible space:

For the violated dimension we compute the new position as if $X_{t+1} > X_{max}$, then

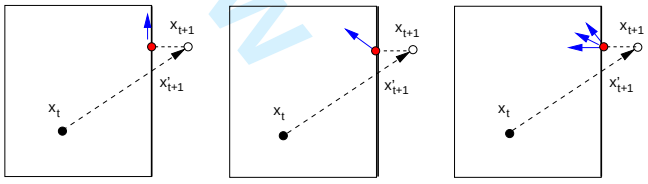
$$X'_{t+1} = X_{max} - (X_{t+1} - X_{max}) \quad (5)$$

if $X_{t+1} < X_{min}$, then

$$X'_{t+1} = X_{min} + (X_{min} - X_{t+1}) \quad (6)$$

Finally, infeasible position vector components can be set to a random position inside the respective bounds $X_{i,min}$ and $X_{i,max}$, denoted as **Random** in the following.

In any case, if just the position is modified, then due to the momentum based on the current velocity, it is very likely that the particle would leave the feasible region in the next iteration again. Therefore, [12], [13], [14] proposed to also modify the *velocity* of such particles. Figure 2 illustrates three ideas for modifying the velocity values in combination with the Nearest strategy.



(a) Absorb (b) Deterministic back (c) Random back

Fig. 2. Different methods to modify velocity settings for a particle relocated onto the border.

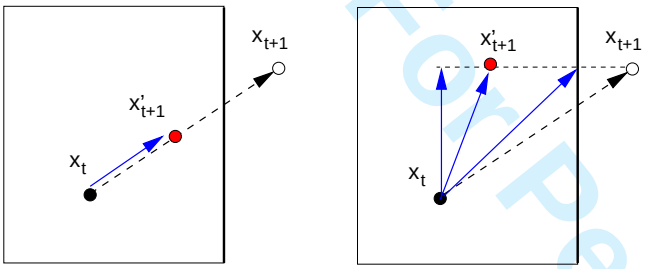
(a) **Absorb/Zero:** The velocity is set to zero: $V_{i,t+1} = 0$
(b) **Deterministic back:** The velocity is reversed to force the particle back into the feasible space in the next iteration: $V_{i,t+1} = -\lambda V_{i,t+1}$. In [13], λ is set to 0.5, in [12] it is set dependent on the particle's distance to the boundary.
(c) **Random back:** If a particle exceeds the boundary in dimension i, the respective velocity vector component is

reversed and multiplied with a random value: $V_{i,t+1} = -\lambda V_{i,t+1}$, where λ is drawn uniformly at random in $[0, 1]$ at each occurence [13].

(d) **Adjust:** After the bound handling procedure, each particle i's velocity vector is set to $\vec{v}_{i,t+1} = \vec{x}_{i,t+1} - \vec{x}_{i,t}$ (used in, e.g., [4]).

C. Preventing infeasibility

Rather than allowing the particle to leave the feasible area and moving it back, other approaches in the literature modify the velocities of the particles such that the new position is always inside the feasible region (c.f., Figure III-C). Because the velocity is modified directly rather than the particle's position, there is no need to adjust the velocity afterwards.



(a) hyperbolic (b) random forth

Fig. 3. Boundary handling methods which always keep the particles inside the feasible area.

In the **Hyperbolic** approach [13], the new velocity $V_{i,t+1}$ is normalized. If $V_{i,t+1} > 0$

$$V_{i,t+1} = \frac{V_{i,t+1}}{1 + |\frac{V_{i,t+1}}{X_{i,max} - X_{i,t}}|} \tag{7}$$

else

$$V_{i,t+1} = \frac{V_{i,t+1}}{1 + |\frac{V_{i,t+1}}{X_{i,t} - X_{i,min}}|} \tag{8}$$

Basically, the closer the particle gets to the boundary (e.g., $X_{i,t}$ only slightly smaller than $X_{i,max}$), the more difficult it becomes to reach it. In fact, the particle is never allowed to reach the boundary completely.

The **Random Forth** approach, also known as random method [13], alters the velocity such that the particle is moved somewhere on the line between the old position and the boundary:

If $X_{i,t+1} > X_{i,max}$, then

$$V_{i,t+1} = rand(0, X_{i,max} - X_{i,t}) \tag{9}$$

if $X_{i,t+1} < X_{i,min}$, then

$$V_{i,t+1} = rand(0, X_{i,t} - X_{i,min}) \tag{10}$$

D. Other approaches

Zhang et al. [15] do not change any of the particle characteristics, but simply extend the search space by an infinite amount of copies. This is visualized for two dimensions in Figure 4, and basically boils down to, for evaluation, mapping the particle back to the original search space by a modulo operation, but otherwise simply ignoring the boundaries. One problem of this approach is that at the boundaries, discontinuities are likely. We call this approach **Periodic**.

$$f(X) = f(X') \quad \text{where} \tag{11}$$

$$X'_i = X_{i,min} + (X_i \text{ MOD } (X_{i,max} - X_{i,min})) \tag{12}$$

IV. Bound Handling Methodologies under Consideration

For the following experimental study, 10 complementing bound handling methods were chosen from the broad variety of available methods: **Hyperbolic** and **RandomBack** performed best in different scenarios in a study by Clerc [13], and **Infinity** was proposed by Bratton and Kennedy as standard method [16]. As velocity clamping (i.e., restricting the maximal velocity) can improve this method [17], Infinity with velocity clamping, denoted as **Infinity-C**, was included. Three repositioning strategies, **Nearest** and **Random**, and **Reflect** were selected, and combined with three different velocity handling methods: (1) **Absorb/Zero**, (2) **Adjust**, and (3) **Unmodified** (an infeasible particle's velocity is not altered).

We furthermore propose a new approaches: In the **Bounded Mirror** method, the search space is expanded similar to the **Periodic** method. However, we mirror the function, to avoid discontinuities. Futhermore, the search space is only doubled in each dimension, and reconnected in a torus-like fashion (i.e., a particle exceeding the upper bound re-enters at the lower bound). For two dimensions, the idea is visualized in Figure 5.

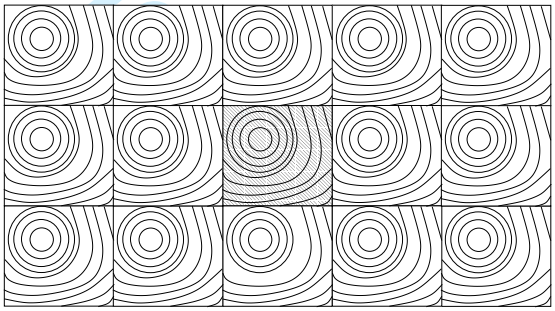


Fig. 4. An example of a two dimensional search space shown in dashed area and the copies of that proposed in the Periodic method from [].

$$X'_{i,t+1} = X_{i,min} + X_{i,t+1} \text{ MOD } (2 \cdot X_{i,max} - X_{i,min})) \tag{13}$$

$$f(X'_{i,t+1}) = \tag{14}$$

$$\begin{cases} f(X'_{i,t+1}) & : \quad X_{i,min} \leq X'_{i,t+1} \leq X_{i,max} \\ f(2 \cdot X_{i,max} - X'_{i,t+1}) & : \quad X_{i,max} \leq X'_{i,t+1} \leq 2X_{i,max} \end{cases}$$

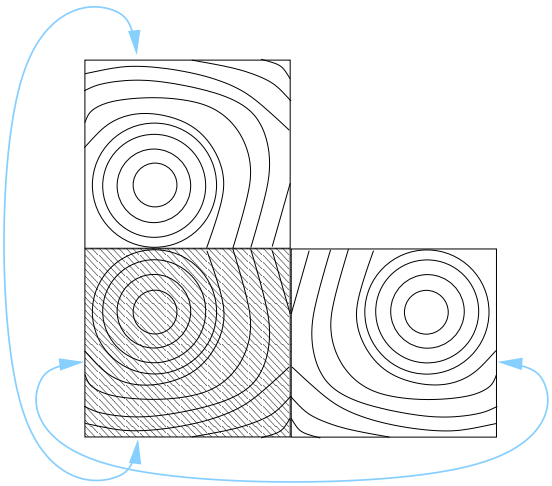


Fig. 5. Bounded Mirror method

V. Experimental Setup

Throughout this paper, we use a standard PSO algorithm with velocity and position update as described in Eqs. (1) and (2). Most of the parameters were chosen according to the values proposed in the *Standard PSO* of Bratton and Kennedy [16]. The parameters in the PSO equations were set to $c_1 = c_2 = 2.05$ and $\chi = 0.72984$.

It is well-known that the particles' neighborhood structure strongly influences the swarm's behavior [19], [20]. A fully connected swarm often suffers from premature convergence, as all particles follow the same local guide. Another popular choice is the *Ring* topology, in which each particle has exactly two neighbors. This structure significantly slows down the swarm's convergence, and might be too slow if function evalutations are expensive. Experimental results published by Kennedy and Mendes [21] indicate that the so-called *von Neumann* topology, which arranges the particles in a two-dimensional grid torus (i.e., with wrap-around edges), can be a good compromise between the fast converging, fully connected swarm and the slow *Ring* topology. So, we use a population of $m = 49$ particles, arranged in a in a 7×7 grid torus. A particle is included in its own neighborhood, i.e., each particle actually has five neighbors: the ones below, above, to the right, to the left, and itself.

VI. Flat Landscape Analysis

Ideally, a bound handling method should keep the search contained within the feasible area, but it should not bias or influence the search within the feasible area, i.e., it should not divert the search away from the boundary, nor should it attract the search towards the boundary. Of course, in some applications, where it is known that the best solutions lie on the boundary of the feasible area, it may be beneficial to bias search towards the boundary, but without such problem specific knowledge, we assume that the absence of any bias would be preferable.

In this section, we analyze the bias a bound handling method imposes on search. The idea is that on a completely flat landscape, the probability of a particular point being sampled by PSO should be equal for all feasible points. Thus, looking at the distribution of samples in different areas and comparing it to the uniform distribution allows us to detect an inherent bias.

Note that a standard PSO where a particle updates its personal best only if it found a *better* solution would never update the information on personal and neighborhood best on a flat landscape and thus not be able to move around. This may be an issue not only in our flat landscape experiments, but also in fitness landscapes with large plateaus or neutral networks. Owen and Harvey [22] have thus suggested to update the local or global best whenever a new solution is found that is *at least as good* as the previous one, resulting in a particle's personal best always being equal to the particle's current position on the flat fitness function. While this may be sensible for maintaining diversity during optimization, we decided that it would too much interfere with the usual particle dynamics. Our approach is thus to make a probabilistic selection whenever fitness values are identical. More specifically, if the particle's current position is as good as its personal best, the new position replaces the previous personal best with 50% probability. Similarly, when there are multiple equally good positions which could be used as neighborhood best, each particle in each iteration selects one of these randomly. In other words, on the flat landscape and with the grid neighborhood we are using, the neighborhood best is chosen uniformly random from the particle's own personal best and its neighbors' personal bests.

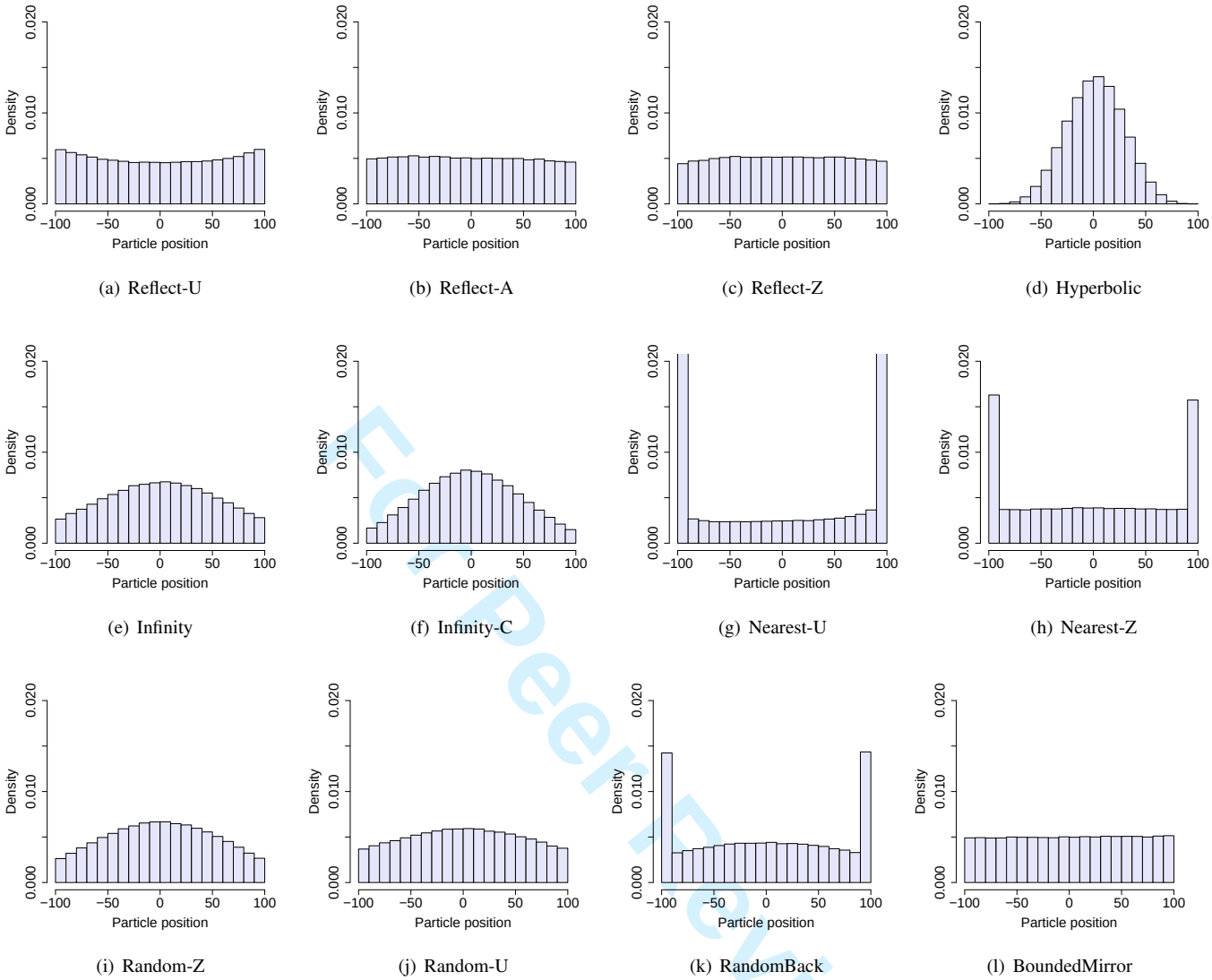
A. Setup

We conducted the experiment with a PSO using 49 particles on a 7x7 grid neighborhood as throughout the paper. The flat landscape has 30 dimensions, each ranging from -100 to 100. Each PSO was run for 300,000 function evaluations, and we replicated each run 10 times. For analysis, each dimension was divided into 20 equally sized intervals, and for each interval, we recorded how often an individual with a coordinate in this interval has been evaluated. Note that the BoundedMirror method actually increases the search space. We map all samples in the "virtual" created search spaces back to the original search space, as this is where these particles are actually evaluated.

B. Results and Discussion

Figure 6 shows the distribution of samples in the different intervals for one run in one arbitrarily chosen dimension of the search space. The other dimensions and runs look almost identical. We have one plot for each bound handling method.

Generally, we can observe that the method to determine the new location has a much larger impact than the way to set the velocity. Figure 6 (a)-(c) shows the distribution for the *Reflect* method with different ways to re-set the velocity. Without changing the velocity (*Reflect-U*), there is a slight



(a) Reflect-U (b) Reflect-A (c) Reflect-Z (d) Hyperbolic

(e) Infinity (f) Infinity-C (g) Nearest-U (h) Nearest-Z

(i) Random-Z (j) Random-U (k) RandomBack (l) BoundedMirror

Fig. 6. Distribution of evaluated samples on the flat landscape in dimension one, depending on the bound handling method used.

bias towards the boundaries of the search space. This is not surprising, as a particle, when it moved out of the search space and got reflected, will likely move towards the boundary again in the next iteration, due to the momentum based on an unaltered velocity. On the other hand, when the velocity is adapted (*Reflect-A*), or set to zero (*Reflect-Z*), what results is a minimal bias towards the center. When looking at the *Nearest* method (Figure 6 (g)-(e)) we see a very strong bias towards the boundary, as it sets an infeasible particle onto the boundary. And again, the velocity setting has only a small effect, with a slight increase in bias towards the boundary when the velocity is not changed (*Nearest-U*), while setting the velocity to zero leads to an even distribution anywhere else (*Nearest-Z*). For *Random*, there is a significant bias towards the center, as particles near the boundary are likely to move outside at some point are are then re-initialized. Once more, not changing the velocity (*Random-U*) favors the boundary (counteracting the center-bias of *Random*), while *Random-Z* and *Random-A* (not shown) are almost identical and have a stronger bias towards the center. The distribution of *Infinity* looks almost identical to *Random-Z*, with a rather strong bias towards the center. Note, however, that for *Infinity*, many invalid individuals are not evaluated and don't show up in this plot. Clamping the velocity reduces the fraction of infeasible solutions produced, but, as Figure 6(f) shows, the bias towards the center is even stronger. The strongest bias towards the center is produced by *Hyperbolic* (Figure 6(d)), Due to the design of the method, it is impossible to move completely onto the boundary. But the results suggest that it is furthermore very hard even to find a solution close to the boundary. *RandomBack* (Figure 6(k)) sets the infeasible particle onto the boundary, hence a bias towards the boundary. But it also sets the velocity away from the boundary, leading to a lesser bias towards the center for the rest of the search space. The only method not showing any

bias on the flat landscape is *BoundedMirror*.

Overall, there are four methods which exhibit a bias towards the center. *Hyperbolic* has the strongest bias towards the center. The second highest bias is caused by *Infinity*, then *Random* and finally, it is barely noticable for *Reflect*. Two methods exhibit a strong bias towards the boundary, as they set any infeasible particle exactly onto the boundary: Nearest-Z and RandomBack. Nearest-Z thereby depicts a more or less even distribution in the non-boundary intervals, while for RandomBack, there is a visible bias among the non-boundary intervals towards the center. This would indicate that RandomBack makes it difficult to find solutions close to, but not on, the boundary. The velocity setting mechanism has a much smaller influence on the bias. But in general, it can be said that keeping the velocity (-U) favors the boundary.

Overall, the flat landscape analysis suggests that the most promising bound handling methods are BoundedMirror and Reflect-Z. If one expects the optima to lie on the boundary or close to the boundary, Nearest-Z seems to be a good strategy.

VII. Benchmark Analysis

In the following experimental analysis, the effect of different bound handling strategies on particle swarm performance is investigated. A wide range of commonly-used benchmark functions is used for this purpose. The experimental study is divided into three parts: As already mentioned in Section IV, many bound handling strategies affect both a particle's position and its velocity when it leaves the search space. Hence, we distinguish position handling and velocity handling. In our first experiment, different velocity handling strategies are compared with each other and their influence on particle swarm performance is discussed. In the second and third experiment, the effect of selected bound handling strategies on the solution quality is studied and compared with the results of the flat landscape analysis. The main question is whether the observations of the flat landscape analysis presented in Section VI can be transferred to non-flat problems.

A. Setup and Testfunctions

In order to investigate the influence of bound handling on particle swarm performance and the swarm's exploration behavior, a standard PSO [16] using different bound handling strategies was examined on a variety of well-known benchmark problems. All parameters except the bound handling mechanism were kept fix in the experiments, and set to standard values.

For the experimental study, seven traditional benchmark functions (Sphere, Rosenbrock, Rastrigin, Griewank, Ackley, Michalewicz, and Schwefel), and the CEC 2005 benchmarks f1–f14 [23] were used. From the CEC 2005 benchmarks, all noisy functions and all functions without bounds were excluded. The benchmark problems were investigated in 2, 5, 30, and 100 dimensions. The function descriptions of the seven traditional benchmarks are given in Table I. In most of them, the global optimum is located at the center of the search space. In order to get more meaningful results, it has become good practice to either shift the position of the global optimum, or to use an asymmetric initialization range [16]. In our experiments, the latter method was applied. The initialization ranges as well as the location and the objective value of the global optimal solution are presented together with the function descriptions in Table I. However, when optimizing a CEC 2005 benchmark, particles were initialized uniformly at random in the whole search space, as proposed by their designers [23].

TABLE I
Function descriptions of the traditional n-dimensional benchmarks used in the experimental study. $\mathcal{S}$: Search space, $\mathcal{I}$: Particle initialization space, $\vec{x}^*$: Global optimal solution.

Sphere	$f(\vec{x}) = \sum_{i=1}^{n} x_i^2$ $\mathcal{S} = [-100 \dots 100]^n$, $\mathcal{I} = [50 \dots 100]^n$ $\vec{x}^* = (0, \dots, 0)$, $f(\vec{x}^*) = 0$
Rosenbrock	$f(\vec{x}) = \sum_{i=1}^{n-1} \left(100 \cdot \left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2\right)$ $\mathcal{S} = [-30 \dots 30]^n$, $\mathcal{I} = [15 \dots 30]^n$ $\vec{x}^* = (1, \dots, 1)$, $f(\vec{x}^*) = 0$
Rastrigin	$f(\vec{x}) = 10 \cdot n + \sum_{i=1}^{n} \left(x_i^2 - 10 \cdot \cos\left(2 \cdot \pi \cdot x_i\right)\right)$ $\mathcal{S} = [-5.12 \dots 5.12]^n$, $\mathcal{I} = [2.56 \dots 5.12]^n$ $\vec{x}^* = (0, \dots, 0)$, $f(\vec{x}^*) = 0$
Griewank	$f(\vec{x}) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$ $\mathcal{S} = [-600 \dots 600]^n$, $\mathcal{I} = [300 \dots 600]^n$ $\vec{x}^* = (0, \dots, 0)$, $f(\vec{x}^*) = 0$
Ackley	$f(\vec{x}) = -20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2}\right)$ $- \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos(2\pi x_i)\right) + 20 + e$ $\mathcal{S} = [-32 \dots 32]^n$, $\mathcal{I} = [16, \dots 32]^n$ $\vec{x}^* = (0, \dots, 0)$, $f(\vec{x}^*) = 0$
Michalewicz	$f(\vec{x}) = -\sum_{i=1}^{n} \left(\sin\left(x_i\right) \cdot \left(\sin\left(\frac{i \cdot x_i^2}{\pi}\right)\right)^{2 \cdot m}\right)$ $\mathcal{S} = [0 \dots 3.14]^n$, $\mathcal{I} = [2.355 \dots 3.14]^n$, $m = 10$
Schwefel	$f(\vec{x}) = \sum_{i=1}^{n} \left(-x_i \cdot \sin\left(\sqrt{\lvert x_i \rvert}\right)\right)$ $\mathcal{S} = [-500 \dots 500]^n$, $\mathcal{I} = [-250 \dots 250]^n$ $\vec{x}^* = (420.9687, \dots)$, $f(\vec{x}^*) = -n \cdot 418.9829$

Each optimization run terminates after $300,000$ function evaluations, and each experimental configuration was repeated 100 times. The analysis of the experimental results is based on the following outcomes:

- Mean objective values and standard errors after $300,000$ function evaluations.
- Convergence plots of mean objective values and standard errors.
- The performance of two algorithms A and B was compared by using the one-sided Wilcoxon rank sum test with null-hypothesis $H_0 : F_A(z) = F_B(z)$ (no performance difference between algorithm A and algorithm B), and the one-sided alternative $H_1 : F_A(z) < F_B(z)$ (algorithm A performed better than algorithm B). $F_X(z)$ is the distribution of the results of algorithm X. The significance level was set to $\alpha = 0.01$.

B. Velocity Handling

In Section IV, several commonly used bound handling methodologies were presented. Whenever a particle leaves the search space, the effects on both its position and its velocity have to be considered. We carried out experiments to examine the influence of the velocity update of infeasible particles on particle swarm performance. Therefore, the position update strategies *Random*, *Reflect* and *Nearest* were combined with the velocity update strategies *Zero (abbr.: -Z)*, *Adjust (-A)* and *Unmodified (-U)*, and tested on all 100-dimensional benchmarks. The result of the Wilcoxon rank sum test for this experiment is shown in Table II. On the tested benchmarks, the *Unmodified* strategy cannot compete with the other two variants. Columns 3, 6, and 9 show that not altering the velocity after resetting a particle's position can lead to significant performance losses. On all benchmarks except Rastrigin, *Zero* and/or *Random* either significantly outperformed *Unmodified*, or the obtained solution quality was similar, according to the Wilcoxon rank sum test. Using the *Unmodified* strategy resulted in noticably poor average objective values (not presented here) in comparison to *Zero* and *Adjust* on many of the investigated testfunctions.

The first table shows that using Nearest-Z often leads to significantly superior results than the application of Nearest-A or Nearest-U. The latter two variants have the following drawback: Suppose that in the d-th search space dimension, the upper bound was violated by particle i in iteration t but not in iteration $t-1$. Then, if the velocity is not altered or if it is adjusted, the d-th component of particle i's velocity vector, $v_{i,t,d}$, is greater than zero, and the particle is again attracted to infeasible regions in the subsequent iteration step. Previous theoretical studies showed that particles often leave the search space as soon as in the first iteration [24]. If there are good local optima on the boundary, eventually all particles might converge there, in particular, if a densely connected neighborhood topology is used. The phenomenon of premature convergence on the boundary was already experimentally observed and discussed earlier [15], [4]. Our flat landscape analysis in Section VI showed that when using the Nearest-Z strategy, the search is biased towards the boundary as well. However, it seems that setting the critical velocity components to zero can help particles to escape boundary regions if necessary.

We already discussed that the *Unmodified* strategy was clearly outperformed by *Zero* and *Adjust* in our experiments, and that Nearest works best in combination with *Zero*. Moreover, Table II shows that *Zero* can also be slightly preferred over *Adjust* when using *Random* or *Reflect* position handling. Therefore, in the subsequent experiments, *Nearest*, *Reflect* and *Random* position handling were used in combination with the *Zero* strategy.

C. Shifted Sphere Analysis

The flat landscape analysis presented in Section VI showed that each bound handling method is biased towards specific areas of the search space, e.g., *Nearest-Z* prefers boundary regions whereas *Random-Z* favors the center. In the following experimental analysis, we investigate whether this bias can also be observed when solving non-flat problems. Therefore, the well-known Sphere benchmark was shifted by different values Y, and the swarm's ability to find optima located at different search space areas was analyzed.

The *Shifted Sphere function*, *SphereY*, has the following function description: $f(\vec{x}) = \sum_{i=1}^{n} (x_i - Y)^2$. The search space $\mathcal{S}$ and the initialization space $\mathcal{I}$ were set to $[-100 \dots 100]^n$. Asymmetric initialization is not used in this case because the function itself is shifted. The problem was examined for 2, 5, 30, and 100 dimensions, and for $Y = 0, 10, 20, \dots, 80, 90, 99, 100$. In the following, we state that a particle swarm has *solved* the Shifted Sphere problem if the obtained objective value was less than 10^{-5}.

First irregularities occur already in the 2-dimensional case: As shown in Table III, the PSO algorithm sometimes did not succeed in solving the Sphere99 problem when using the *Nearest-Z* strategy. In the flat landscape analysis, we observed that *Nearest-Z* is biased towards the search space boundary. The reason for its inability to reliably solve the Sphere99 problem is that the swarm sometimes gets stuck on the boundary. The situation can be even worse if velocities are not set to zero, as outlined in the previous section and mentioned in earlier results [15], [4]. However, it might be a good idea to pull particles back into the search space. When using *RandomBack*, particles never converged on the boundary in our experimental analysis on the Shifted Sphere functions, although it is biased towards the boundary on a flat landscape.

In the flat landscape analysis, two bound handling methods were identified to introduce a significant center bias into the PSO algorithm: *Hyperbolic* and *Random-Z*. Hence, we would expect these methods to successfully solve a Shifted Sphere problem if the global optimum is located in or near the search space center, but to have difficulties to provide good solutions if the global optimum is placed near the boundary. However, although *Hyperbolic* is strongly biased towards the center when solving the flat landscape problem, it does not fail to approach good solutions near the boundary. The results obtained when using this methodology are satisfactory, even for Sphere99 and Sphere100, and even for high-dimensional problems. Hence, despite the center bias, *Hyperbolic* is able to explore boundary regions. However, the convergence speed can be very slow. Details are shown in the convergence plot in Figure 7: When solving the Sphere100 problem, *Hyperbolic* is able to provide good results only if the algorithm is run for more than approx. 50,000 function evaluations. On the other hand, when solving the Sphere0 or Sphere10 problem, which have their global optimum in or near the center, *Hyperbolic* converges much faster than all other bound handling strategies (see Figure 8). These convergence analyses confirm the center bias of *Hyperbolic* observed when solving the flat landscape problem.

In contrast to *Hyperbolic*, *Random-Z* is not able to approach good solutions at the search space boundary when solving high-dimensional problems. Significant performance losses

TABLE II

SUMMARY OF ONE-SIDED WILCOXON RANK SUM TEST WITH SIGNIFICANCE LEVEL 0.01. FOR EACH ALGORITHMIC COMBINATION (A, B), THIS MATRIX SHOWS HOW OFTEN A PERFORMED SIGNIFICANTLY BETTER THAN B. *Example:* ENTRY 9 IN THE FIRST TABLE SHOWS THAT NEAREST-Z SIGNIFICANTLY OUTPERFORMED NEAREST-U (3) ON 9 BENCHMARKS. THE TOTAL NUMBER OF BENCHMARKS IS 19. THE VELOCITY HANDLING STRATEGIES ARE ABBREVIATED: -Z DENOTES *Zero*, -A DENOTES *Adjust*, AND -U DENOTES *Unmodified*.

	1	2	3
Nearest-Z (1)	0	8	9
Nearest-A (2)	1	0	8
Nearest-U (3)	0	0	0

	4	5	6
Random-Z (4)	0	5	10
Random-A (5)	1	0	10
Random-U (6)	1	1	0

	7	8	9
Reflect-Z (7)	0	2	10
Reflect-A (8)	0	0	10
Reflect-U (9)	1	1	0

TABLE III

AVERAGE VALUES AND STANDARD ERRORS OF THE EXPERIMENTAL ANALYSIS ON SPHERE99 AND SPHERE100, IN 2-, 5-, AND 30-DIMENSIONS.

	2D Sphere99	**5D** Sphere99	**30D** Sphere99
Hyperbolic	4.3345e-12±4.6874e-13	6.412e-09±2.9573e-10	9.657e-07±1.185e-08
RandomBack	**3.4953e-12±3.8284e-13**	6.4054e-09±2.8868e-10	9.7198e-07±1.2917e-08
Nearest-Z	*0.1±0.030151*	*0.2±0.04264*	0.52±0.097938
Random-Z	4.3011e-12±4.6263e-13	6.8431e-09±3.1012e-10	*504.42±12.418*
Reflect-Z	3.655e-12±3.5758e-13	**6.224e-09±2.5881e-10**	9.5121e-07±1.5062e-08
BoundedMirror	8.6854e-12±8.7458e-13	7.0681e-09±3.1692e-10	9.7384e-07±1.4381e-08
Infinity	3.9552e-12±3.9737e-13	6.656e-09±2.9343e-10	**8.9325e-07±1.1927e-08**
Infinity-C	4.4446e-12±4.8675e-13	6.5404e-09±2.9843e-10	9.5927e-07±1.3301e-08
	2D Sphere100	**5D** Sphere100	**30D** Sphere100
Hyperbolic	6.4756e-09±3.4944e-10	4.8356e-08±2.3166e-09	6.9615e-07±3.8906e-08
RandomBack	**0±0**	**0±0**	4.5013e-07±7.9305e-09
Nearest-Z	**0±0**	**0±0**	**2.4847e-08±4.3738e-09**
Random-Z	*2.5553e-04±3.5961e-05*	*0.55797±0.026257*	*905.46±16.188*
Reflect-Z	1.835e-12±1.8805e-13	2.7189e-09±1.1198e-10	4.2589e-07±5.5026e-09
BoundedMirror	4.8199e-12±5.1443e-13	6.1907e-09±2.8834e-10	9.6749e-07±1.515e-08
Infinity	1.0273e-11±1.0907e-12	1.7084e-08±5.9589e-10	1.4779e-06±1.5638e-08
Infinity-C	1.2462e-11±1.3982e-12	1.7653e-08±6.8277e-10	1.3151e-06±1.3195e-08

can already be noticed when solving Sphere99 or Sphere100 in 30 dimensions (see Table III), and are even more obvious when solving the respective 100-dimensional problems (see Table IV and Figure 7).

Considering the flat landscape analysis, there are two clear winners: When using *Reflect-Z*, the particles are only slightly distracted from the boundary, and they uniformly explore the flat landscape when using *BoundedMirror* bound handling. Both *Reflect-Z* and *BoundedMirror* were able to solve the Shifted Sphere benchmarks regardless of the value of Y, and for every investigated dimensionality (the obtained average values are shown in Tables III and IV). However, when applying *BoundedMirror*, the particle swarm mostly converged very slowly and unpredictable in high-dimensional spaces: When solving the 100-dimensional Sphere100 problem, solutions of very good quality were not available until approx. 150,000 function evaluations had passed, and a similar behavior can also be observed for Sphere0. See Figures 7 and 8 for details. Although the flat landscape is almost ideally explored, *BoundedMirror* is not the best choice for high-dimensional spaces probably due to the fact that the search space volume is strongly increased, by a factor of 2^n. Using *Reflect-Z* does not have this disadvantage. Instead, the particles converged fast, and delivered results of very good quality throughout the Shifted Sphere benchmark set.

Summarized, the Shifted Sphere experiment confirmed the results derived from our flat landscape analysis presented in Section VI, e.g., the center bias of *Hyperbolic* and *Random-Z*, the swarm's affinity towards the boundary when using *Nearest-Z*, and that the swarm is nearly unbiased when applying *Reflect-Z* bound handling. Furthermore, we learned that *BoundedMirror* might not be suited for high-dimensional problems due to the significant increase of the search space volume, although it performed best in the flat landscape analysis.

D. Benchmark Analysis

Both the flat landscape and the Shifted Sphere function are rather simple optimization problems. This last experiment is dedicated to the question which bound handling strategies are suited for which class of problems in order to provide hints for practical PSO application. In particular, we are interested if the results of the flat landscape and the Shifted Sphere analysis apply for other problems as well. Summarizing the results of the flat landscape and the Shifted Sphere analysis, there is a clear winner: the *Reflect-Z* strategy. Hence, we are especially interested in the question how *Reflect-Z* performs in comparison with the other methodologies.

In order to answer these questions, seven well-known traditional benchmark functions (Sphere, Rosenbrock, Rastrigin, Griewank, Ackley, Michalewicz, and Schwefel, see Table I), and the CEC 2005 benchmarks f1–f14 [23] were used as

TABLE IV
AVERAGE VALUES AND STANDARD ERRORS FOR THE EXPERIMENTAL ANALYSIS ON THE SHIFTED SPHERE BENCHMARKS (100-DIMENSIONAL).

	Sphere	Sphere10	Sphere20	Sphere30
Hyperbolic	6.1045e-06±9.3119e-08	6.0733e-06±9.7688e-08	**5.9109e-06±9.6418e-08**	6.0458e-06±8.7989e-08
RandomBack	5.9764e-06±8.4248e-08	5.9914e-06±8.6473e-08	6.1546e-06±1.0511e-07	6.0627e-06±9.909e-08
Absorb	6.2e-06±1.061e-07	6.0914e-06±8.7046e-08	6.0497e-06±8.1365e-08	6.0768e-06±8.9161e-08
Random-Z	6.0864e-06±9.4963e-08	6.0021e-06±8.215e-08	6.2209e-06±9.1495e-08	**5.9615e-06±8.6349e-08**
Random	**5.8033e-06±8.0123e-08**	6.1241e-06±8.8645e-08	6.0773e-06±9.3729e-08	6.1626e-06±8.8506e-08
Reflect-Z	5.9754e-06±8.8948e-08	**5.9803e-06±8.3014e-08**	6.0071e-06±9.1625e-08	6.0113e-06±8.8524e-08
Reflect	6.3086e-06±1.1249e-07	6.1255e-06±9.0825e-08	6.163e-06±9.6449e-08	6.0969e-06±9.955e-08
BoundedMirror	6.3577e-06±7.9919e-08	6.3554e-06±2.0867e-07	6.4364e-06±9.2964e-08	6.5288e-06±1.2903e-07
Infinity	*12394±3320.5*	*18436±3777.1*	*25559±4500.2*	*47893±6700.3*
Infinity-C	6.2421e-06±1.0176e-07	6.0626e-06±8.343e-08	6.2103e-06±8.5842e-08	6.0938e-06±8.7021e-08
	Sphere40	Sphere50	Sphere60	Sphere70
Hyperbolic	6.0746e-06±8.8186e-08	**5.7862e-06±7.775e-08**	5.9714e-06±8.4452e-08	6.0507e-06±9.7184e-08
RandomBack	**5.9356e-06±9.0206e-08**	5.8645e-06±9.9475e-08	6.2003e-06±9.9254e-08	6.0492e-06±8.7276e-08
Absorb	6.361e-06±1.7607e-07	6.0637e-06±8.0149e-08	6.145e-06±9.2158e-08	**5.9379e-06±8.4428e-08**
Random-Z	5.9702e-06±9.1828e-08	6.0661e-06±8.4508e-08	**5.9418e-06±8.3288e-08**	6.2312e-06±9.0911e-08
Random	6.1531e-06±9.4541e-08	6.1187e-06±1.0234e-07	6.1291e-06±8.5198e-08	6.1516e-06±8.3979e-08
Reflect-Z	6.0776e-06±8.5007e-08	6.0506e-06±8.9848e-08	5.9772e-06±8.5124e-08	6.3045e-06±1.0124e-07
Reflect	6.1332e-06±1.012e-07	5.9365e-06±8.4451e-08	6.0569e-06±9.8957e-08	5.981e-06±8.7367e-08
BoundedMirror	6.3577e-06±1.1356e-07	8.8568e-05±8.1982e-05	6.3394e-06±1.1244e-07	6.4027e-06±1.2207e-07
Infinity	*80281±8602.9*	*145250±10894*	*212350±12571*	*301670±13765*
Infinity-C	6.0471e-06±8.6956e-08	6.0937e-06±8.7881e-08	5.9662e-06±7.7867e-08	6.0237e-06±8.6457e-08
	Sphere80	Sphere90	Sphere99	Sphere100
Hyperbolic	6.083e-06±1.0817e-07	5.9648e-06±8.7999e-08	5.932e-06±8.0168e-08	3.8653e-06±1.1342e-07
RandomBack	6.1176e-06±1.0538e-07	5.9907e-06±8.6244e-08	5.921e-06±8.9808e-08	2.1426e-06±2.1504e-08
Absorb	4±4	2±1.4071	0.72015±0.11022	**7.7479e-07±1.4735e-08**
Random-Z	6.2173e-06±9.0094e-08	7.028e-06±1.0463e-07	20839±329.47	24484±363.03
Random	6.4698e-06±1.308e-07	10879±250.45	56731±634.04	62532±617.45
Reflect-Z	**5.9744e-06±1.0193e-07**	6.0886e-06±8.9738e-08	6.0368e-06±9.1581e-08	1.526e-06±1.1241e-08
Reflect	6.2391e-06±9.3654e-08	6.0695e-06±8.9505e-08	**5.8621e-06±8.8721e-08**	1.4397e-06±1.0569e-08
BoundedMirror	6.4613e-06±1.2418e-07	6.3467e-06±1.3653e-07	6.3585e-06±8.9263e-08	6.2857e-06±9.7334e-08
Infinity	*487880±12390*	*594410±16177*	*753030±16884*	*756960±15272*
Infinity-C	6.1323e-06±9.5742e-08	**5.7484e-06±8.9259e-08**	165.92±103.47	364.56±93.455

a benchmark set. We excluded all noisy functions and all functions without boundary constraints from the investigation. All problems were tested using a 2, 5, 30, and 100-dimensional parameter space. In the following, we will first present the results, and give an interpretation of these experimental results afterwards.

1) Results: Mostly, the PSO was able to find the global optimum for the 2-dimensional problems, given 300,000 function evaluations. However, there are some exceptions, shown in Table V. The following performance differences are statistically significant, considering the results of the one-sided Wilcoxon rank sum test: *Hyperbolic* was significantly outperformed by all other bound handling variants on Michalewicz, Schwefel, and f8. *Random-Z* was significantly outperformed by all other methods on f5, and by all strategies except *Hyperbolic* on f8.

On many 5-dimensional problems, all particle swarms provided solutions of similar quality. The global optimum of f8 was not found by any of the particle swarm optimizers: Each swarm achieved an equally poor average objective value of approx. -119.99 (global optimum: -140). Again, *Hyperbolic* was significantly outperformed by all other bound handling strategies on Michalewicz and Schwefel, and additionally on Rastrigin, f9, and f14. *Random-Z* was again significantly outperformed by all other strategies on f5. Another remarkable result is that *BoundedMirror* performed significantly better than the other strategies on Schwefel.

For the 100-dimensional problems, results are shown in Table VI. The results of the Wilcoxon rank sum tests are presented in Table VII to show which performance differences are statistically significant. However, note that very many rank sum tests were performed and that there is a high probability that some "significant" results occured simply by chance. We react to this fact by performing very careful interpretation of the results obtained from the Wilcoxon rank sum test, and by considering results only as significant if they are multiply confirmed.

2) Discussion: The results of the flat landscape and Shifted Sphere analysis are confirmed considering the 2- and 5-dimensional benchmarks. In higher dimensions, bound handling becomes more important, and more significant performance differences can be observed.

The CEC benchmarks f5 and f8 were designed such that their global optimum is located on the boundary. In the 2- and 5-dimensional case, all methods except *Random-Z* were able to solve f5 which confirmes the center bias of *Random-Z*. However, like in the Shifted Sphere analysis, *Hyperbolic* was able to approach the global optimum in f5. f8 is a rotated Ackley function with global optimum on the boundary. The original Ackley is designed such that it is very flat in most regions of the search space, providing nearly no hints for a particle swarm optimizer where to search for the global optimal solution. A deep valey in the center of the

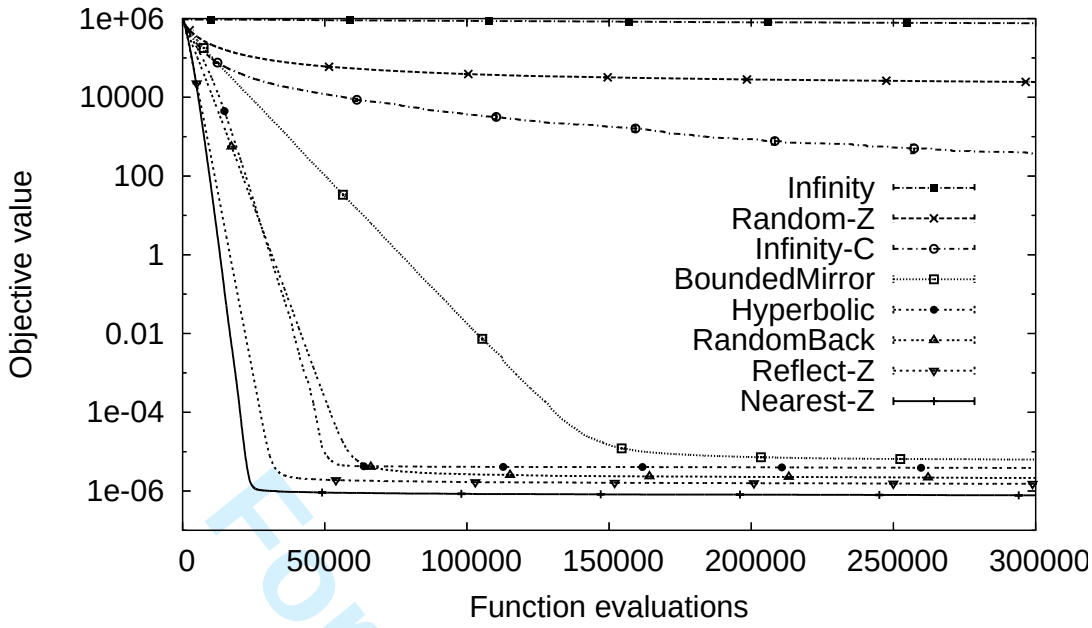


Fig. 7. Average objective values of different bound handling strategies on the Sphere100 function. Vertical bars (very small) show the standard errors.

TABLE V
AVERAGE OBJECTIVE VALUES AND STANDARD ERRORS OF DIFFERENT BOUND HANDLING STRATEGIES ON ALL 2-DIMENSIONAL BENCHMARKS WITH SIGNIFICANT PERFORMANCE DIFFERENCES. THE BEST OBJECTIVE VALUES ARE PRESENTED TOGETHER WITH THE FUNCTION NAME WHERE AVAILABLE.

	Michalewicz	Schwefel (≈ -837.9658)	f5 (-310)	f8 (-140)
Hyperbolic	*-1.8204±5.4679e-03*	*-491.73±8.0999*	**-310±0**	*-121.64±0.47711*
RandomBack	**-1.9498±0**	**-837.97±0**	**-310±0**	**-140±0**
Absorb	-1.9473±1.753e-03	**-837.97±0**	**-310±0**	**-140±0**
Random-Z	-1.9473±1.753e-03	**-837.97±0**	-309.05±0.054523	-139.32±0.34364
Random	-1.9485±1.2459e-03	**-837.97±0**	*-298.48±0.51144*	-138.73±0.20269
Reflect-Z	-1.9485±1.2459e-03	**-837.97±0**	**-310±0**	-139.4±0.34289
Reflect	-1.9436±2.7291e-03	**-837.97±0**	**-310±0**	-139.8±0.2
BoundedMirror	**-1.9498±0**	**-837.97±0**	**-310±0**	**-140±0**
Infinity	**-1.9498±0**	**-837.97±0**	**-310±0**	-139.63±0.23583
Infinity-C	-1.9473±1.753e-03	**-837.97±0**	**-310±0**	-139.2±0.39389

parameter space contains the global optimum. Respectively, f8 has a deep valey at the boundary and is very flat elsewhere. This setting prevented many swarms to approach the global optimum, as shown in Table V. The only successful bound handling strategies were the two methods which are biased towards the boundary according to the flat landscape analysis, *Nearest-Z* and *RandomBack*, and the unbiased *BoundedMirror* methodology. In the higher-dimensional experiments, none of the swarms was able to solve f8, due to its needle-in-the-haystack character.

The higher the problem's dimensionality, the stronger are the performance differences among the different bound handling methods. Due to the high dimensionality and the difficulty of the problems, the biases observed in the flat landscape and Shifted Sphere analysis cannot be confirmed as clearly as in the lower-dimensional experiment. Other effects seem to have a stronger impact on the solution quality. However, the experimental results of the 30- and 100-dimensional problems are interesting for practical PSO application which often have a high-dimensional (bound-)constrained parameter space.

As observed earlier [24], the often-used *Infinity* strategy is not suited for high-dimensional search spaces. The obtained average objective value is often very poor for the 100-dimensional testfunctions (see Table VI). Note, however, that in 30 dimensions, *Infinity* without velocity clamping provided satisfactory results. When solving a high-dimensional optimization problem, many particles are expected to leave the parameter space in the first iteration [24]. Hence, when using *Infinity* bound handling, many particles mainly explore invalid space at the beginning of the optimization process. The use of velocity clamping often improved the average objective value.

Similarly, *BoundedMirror* provided good average solution quality on many 2-, 5-, and 30-dimensional problems, but the performance sometimes strongly deteriorated when solving the 100-dimensional benchmarks. We assume that the reason is the strong increase of the search space volume. When

TABLE VI

AVERAGE VALUES AND STANDARD ERRORS OF DIFFERENT BOUND HANDLING STRATEGIES ON ALL TESTED BENCHMARKS (100-DIMENSIONAL). THE BEST OBJECTIVE VALUES ARE PRESENTED TOGETHER WITH THE FUNCTION NAME.

	Sphere (0)	Rosenbrock (0)	Ackley (0)	Griewank (0)
Hyperbolic	6.0991e-06±9.9879e-08	**170.2±5.7773**	**0.37521±0.065438**	**2.8395e-03±4.8179e-04**
RandomBack	6.0006e-06±1.0042e-07	206.42±6.2185	1.6451±0.078303	4.9698e-03±1.0694e-03
Absorb	6.1612e-06±8.4985e-08	202.86±6.7151	1.7005±0.069327	6.8967e-03±1.4834e-03
Random-Z	6.0223e-06±9.1279e-08	196.26±6.278	1.5459±0.076438	4.6485e-03±1.0456e-03
Random	6.0717e-06±8.6509e-08	195.36±5.9887	1.7119±0.068808	5.4837e-03±1.6729e-03
Reflect-Z	6.1364e-06±9.0257e-08	207.73±5.9272	1.4626±0.085977	3.2046e-03±7.1381e-04
Reflect	6.1975e-06±9.5109e-08	201.05±6.1938	1.6294±0.075987	3.5266e-03±6.8353e-04
BoundedMirror	6.1399e-06±8.8897e-08	198.35±6.3045	2.3821±0.11136	4.8126e-03±1.2434e-03
Infinity	*257.59±202.37*	*225120±176390*	*15.43±0.37338*	*2.341±1.5805*
Infinity-C	**5.8106e-06±8.9932e-08**	186.36±6.4013	2.3523±0.061027	8.8868e-03±2.6095e-03
	Michalewicz	Rastrigin (0)	Schwefel (≈ -41898.3)	f1 (-450)
Hyperbolic	**-86.063±0.18036**	**164.56±2.6617**	*-15355±184.88*	**-450±0**
RandomBack	-78.403±0.32374	312.98±4.2073	-25612±161.9	**-450±0**
Absorb	-78.345±0.29364	342.43±4.3643	-26484±151.24	**-450±0**
Random-Z	-81.793±0.24097	251.75±3.1807	-23921±147	**-450±0**
Random	-82.933±0.23303	236.83±2.5578	-22942±143.9	**-450±0**
Reflect-Z	-78.084±0.29885	330.21±3.5962	**-29606±207.98**	**-450±0**
Reflect	-78.444±0.29046	348.95±4.4523	-29251±166.54	**-450±0**
BoundedMirror	-77.75±0.3707	275.66±3.941	-26953±134.31	**-450±0**
Infinity	*-71.602±0.57711*	*388.76±6.9991*	-18178±90.386	*149330±11743*
Infinity-C	-78.845±0.34489	345.48±4.3513	-20518±143.28	**-450±0**
	f2 (-450)	f3 (-450)	f5 (-310)	f6 (390)
Hyperbolic	**3394.1±58.766**	**11727000±261660**	28219±271.24	**568.29±4.8617**
RandomBack	29278±830.22	36851000±1069300	29944±450.91	590.13±5.6852
Absorb	40695±1187.6	53671000±2144700	29524±429.39	156270±155650
Random-Z	24035±773.47	78978000±3292400	32105±356.57	585.33±5.2145
Random	34040±1412.3	1.5258e+08±8141000	33720±380.52	582.77±5.1349
Reflect-Z	22989±571	36132000±917720	27173±358.98	576.15±5.6217
Reflect	24518±683.91	36889000±1103800	**26681±359.26**	600.06±5.0376
BoundedMirror	26763±657.22	36153000±900690	31919±408.99	598.89±5.6204
Infinity	*1070200±48804*	*8.6207e+09±4.3234e+08*	*93025±1065.9*	*9.6912e+10±8.8793e+09*
Infinity-C	10746±361.05	1.737e+07±452710	34647±412.88	588.1±4.8021
	f8 (-140)	f9 (-330)	f10 (-330)	f11 (-460)
Hyperbolic	**-118.71±2.7837e-03**	260.28±5.0671	495.27±7.8779	**214.81±0.72584**
RandomBack	**-118.71±2.9245e-03**	36.693±4.8195	102.57±7.8605	223.91±0.62096
Absorb	**-118.71±3.2058e-03**	43.252±5.2026	43.263±7.3072	222.77±0.52507
Random-Z	-118.7±2.6876e-03	53.346±5.3871	89.885±6.5438	222.67±0.60759
Random	-118.69±2.7673e-03	72.444±5.8994	129.2±7.3471	223.94±0.57508
Reflect-Z	**-118.71±2.8503e-03**	**-2.547±6.0854**	**-0.7819±5.6275**	219.28±0.61104
Reflect	**-118.71±3.0183e-03**	-0.70505±5.2413	4.8109±5.5037	218.94±0.6173
BoundedMirror	-118.7±2.8472e-03	63.244±6.5548	173.63±7.7015	225.26±0.54119
Infinity	*-118.6±4.1345e-03*	*1025.9±30.503*	*1867.9±33.698*	*257.65±0.72128*
Infinity-C	-118.7±3.221e-03	229.29±7.5595	288.92±11.142	224.15±0.60764
	f12 (90)	f13 (-130)	f14 (-300)	
Hyperbolic	**184130±11227**	**-102.8±0.51827**	**-254.1±0.059543**	
RandomBack	575560±23888	-60.212±1.7658	-253.25±0.041769	
Absorb	1100900±47792	-66.149±1.6295	-253.37±0.046871	
Random-Z	368380±17310	-66.064±1.4776	-253.41±0.040798	
Random	727850±41578	-64.505±1.5303	-253.47±0.045182	
Reflect-Z	604740±28252	-65.677±1.5125	-253.53±0.046329	
Reflect	671590±28547	-67.248±1.4701	-253.44±0.038034	
BoundedMirror	435700±17925	-40.451±2.7269	-253.08±0.039463	
Infinity	*3.278e+07±928470*	*506500±85463*	*-251.67±0.05942*	
Infinity-C	275740±12189	-65.801±1.3066	-253.59±0.046806	

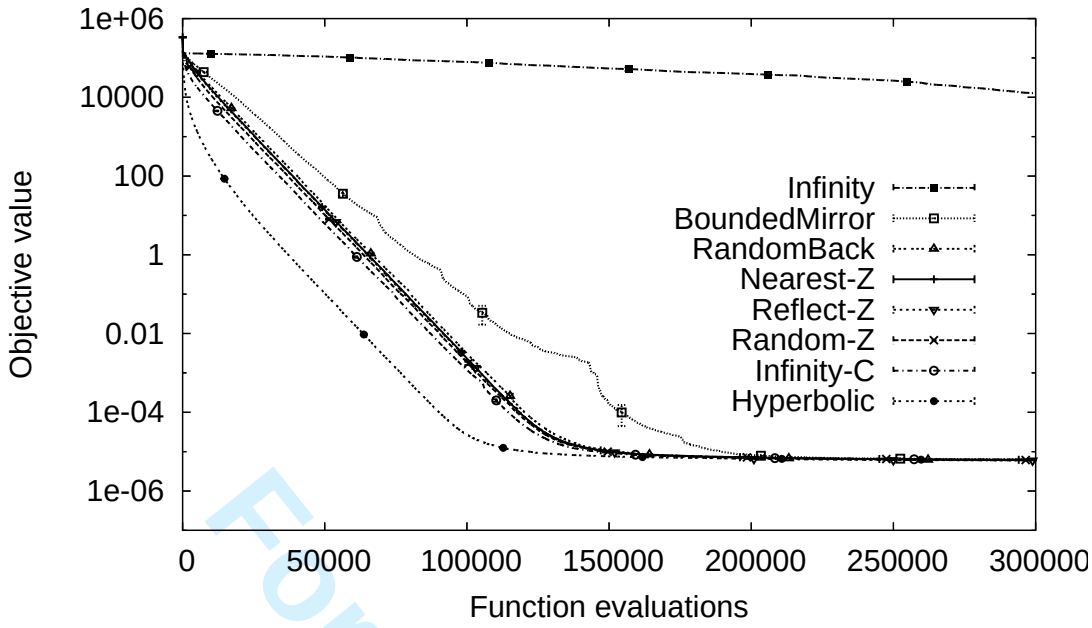


Fig. 8. Average objective values of different bound handling strategies on the Sphere0 function. Vertical bars show the standard errors.

using *BoundedMirror*, the search space volume is increased by a factor of 2^n, where n is the problem's dimensionality. Hence, although *BoundedMirror* performed perfectly on the flat landscape, it has some serious drawbacks when applied on high-dimensional problems.

The flat landscape analysis showed that both *Nearest-Z* and *RandomBack* are biased towards the boundary. Furthermore, the Shifted Sphere experiments showed that particle swarms using *Nearest-Z* bound handling are in danger of prematurely converging on the boundary. In the experimental study on the 30- and 100-dimensional benchmarks both methods only showed mediocre performance. E.g., they cannot compete with *Reflect-Z* on the 30- and 100-dimensional benchmarks considering the results of the Wilcoxon rank sum tests (Table VII). Moreover, when using *Nearest-Z*, the obtained average obtained objective value sometimes is very poor, with high standard error, e.g., when solving the 100-dimensional CEC 2005 benchmark f6.

We already learned that *Random-Z* often is not able to approach boundary solutions. This finding is also confirmed by the experimental results on the 30- and 100-dimensional problems. Considering average values and the results of the Wilcoxon rank sum test, *Random-Z* performed very good on the traditional benchmarks which have their global optimum in or near the search space center (Sphere, Rosenbrock, Ackley, Griewank, and Rastrigin). However, the use of *Random-Z* is disadvantageous for problems which have their good solutions on or near the boundary (Schwefel, f5).

Hyperbolic performed exceptionally well on most of the 30- and 100-dimensional problems, whereas its performance on the 2- and 5-dimensional problems is only average. On the 100-dimensional testbed, *Hyperbolic* significantly outperformed all other bound handling strategies on Ackley, Michalewicz, Rastrigin, f2, f3, f11, f12, f12, and f14 (see Table VII). The obtained average objective values are outstandingly good for most functions (see Table VI). However, *Hyperbolic* performed very poorly on Schwefel, f9, and f10 (see Table VI). The Shifted Sphere analysis showed that *Hyperbolic* is able to approach boundary regions despite its strong bias towards the search space center. This finding is confirmed by the benchmark experiments: *Hyperbolic* achieved good average objective values when solving f5, which has its global optimum on the boundary. Hence, the reason for its bad performance of Schwefel, f9, and f10 must be different. The CEC benchmarks f9 and f10 have a huge number of local optima [23]. We meassured the average distance a particle covered for all bound handling strategies and functions. The results for a representative set of 100-dimensional problems are depicted in Table VIII. We notice that the velocities when using *Hyperbolic* are relatively small. Hence, the exploration capabilities of swarms using *Hyperbolic* is diminished, and it is more probable that the swarm prematurely converges on a local optimum. Although *Hyperbolic* provided extremely good solutions for most high-dimensional problems, there are problems which could not be solved satisfactory. *Hyperbolic* seems to have difficulties with highly multimodal functions. For high-dimensional problems with a moderate number of local optima that are known not to lie on the boundary, *Hyperbolic* might be a good choice.

The last bound handling strategy under consideration is *Reflect-Z*. On a flat landscape, this method was only very slightly biased towards the search space center, and it had no difficulties with solving the Shifted Sphere benchmark. Swarms using *Reflect-Z* showed good performance throughout

TABLE VII
SUMMARY OF ONE-SIDED WILCOXON RANK SUM TEST WITH SIGNIFICANCE LEVEL 0.01 FOR THE 100-DIMENSIONAL BENCHMARKS. FOR EACH ALGORITHMIC COMBINATION (A, B), THIS MATRIX SHOWS ON WHICH BENCHMARKS A PERFORMED SIGNIFICANTLY BETTER THAN B. $\mathcal{B}$ DENOTES THE SET OF ALL 19 BENCHMARKS.

	1	2	3	4	5	6	7	8
Hyperbolic (1)	{}	{Ro, Ack, Mich, Rastr, f2, f3, f5, f6, f11, f12, f13, f14}	{Ro, Ack, Mich, Rastr, f2, f3, f5, f6, f11, f12, f13, f14}	{Ro, Ack, Mich, Rastr, f2, f3, f5, f11, f12, f13, f14}	{Ro, Ack, Mich, Rastr, f2, f3, f11, f12, f13, f14}	{Ro, Ack, Mich, Rastr, f2, f3, f5, f6, f8, f11, f12, f13, f14}	$\mathcal{B}\setminus$ {Sphere, Ro, Schwefel}	{Ack, Mich, Rastr, f2, f3, f5, f6, f8, f11, f12, f13, f14}
Random Back (2)	{Schwefel, f9, f10}	{}	{Rastr, f2, f3, f12}	{Schwefel, f3, f5}	{Rastr}	{Ack, f5, f9, f10, f13, f14}	$\mathcal{B}\setminus$ {Ro}	{Ack, Rastr, Schwefel, f5, f8, f9, f10}
Nearest-Z (3)	{Schwefel, f9, f10}	{Schwefel, f10, f13}	{}	{Schwefel, f3, f5, f10}	{}	{Ack, f5, f10, f11, f13, f14}	$\mathcal{B}\setminus$ {Sphere, Ro}	{Ack, Schwefel, f5, f8, f9, f10}
Random-Z (4)	{Schwefel, f9, f10}	{Mich, Rastr, f2, f12, f13, f14}	{Mich, Rastr, f2, f12}	{}	{Mich, Rastr, f12}	{Ack, Mich, Rastr, f2, f10, f11, f12, f13, f14}	$\mathcal{B}\setminus$ {Ro}	{Ack, Mich, Rastr, Schwefel, f5, f9, f10}
Reflect-Z (5)	{Schwefel, f5, f9, f10}	{Schwefel, f2, f5, f9, f10, f11, f13, f14}	{Schwefel, f2, f3, f5, f6, f9, f10, f11, f12, f14}	{Schwefel, f3, f5, f9, f10, f11}	{}	{Ack, Schwefel, f2, f5, f6, f8, f9, f10, f11, f13, f14}	$\mathcal{B}\setminus$ {Sphere, Ro}	{Ack, Rastr, Schwefel, f5, f8, f9, f10, f11}
Bounded Mirror (6)	{Schwefel, f9, f10}	{Rastr, Schwefel, f12}	{Rastr, f2, f3, f12}	{Schwefel, f3}	{Rastr, f12}	{}	$\mathcal{B}\setminus$ {Sphere, Ro}	{Rastr, Schwefel, f5, f9, f10}
Infinity (7)	{Schwefel}	{}	{}	{}	{}	{}	{}	{}
Infinity-C (8)	{Sphere, Schwefel, f9, f10}	{f2, f3, f12, f14}	{Sphere, f2, f3, f12, f14}	{f2, f3, f12, f14}	{Sphere, f2, f3, f12}	{Sphere, f2, f3, f12, f13, f14}	$\mathcal{B}\setminus$ {Ro}	{}

the given benchmark set. Although it was often significantly outperformed by *Hyperbolic*, there is, in contrast to *Hyperbolic*, no function where *Reflect-Z* performed extremely bad in comparison to the other bound handling strategies.

In this section, we presented a broad experimental study of different bound handling strategies on a wide range of commonly-used benchmark functions. The benchmark analysis confirmed the results obtained in the flat landscape and Shifted Sphere analysis. Furthermore, we identified specific advantages and drawbacks of the different strategies. From the investigated methods, we recommend *Reflect-Z* if nothing is known of the optimization problem beforehand. *Reflect-Z* showed satisfactory performance throughout the whole benchmark set in comparison to the other bound handling strategies.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we have examined various bound handling techniques for PSO. As we have shown, the bound handling technique has a huge impact on the performance of the PSO, especially if the number of dimensions of the search space is high. Compared to EAs, bound handling is more difficult, as particles have a momentum, and the velocity has to be set appropriately. Based on an analysis of flat landscapes, we were able to demonstrate that most bound handling techniques proposed in the literature introduce some bias into the search. Such a bias can be used intentionally, e.g., if it is known that the optimum will be on or close to the boundary. In the absence of such a priori knowledge, however, we would recommend a technique without bias. To evaluate the performance of various bound handling techniques in optimization, we tested them on a large variety of test problems, including a simple sphere with the optimum at various distances from the boundary, and the CEC benchmark test suite. Based on the analysis, it seems the *Reflect-Z* method is the overall best performing candidate. It reflects the particle's movement at the boundary, and sets the velocity to zero. Another good technique is *Hyperbolic*, although it seems to struggle with highly multimodal landscapes. Contrary to common belief, the *Infinity* method, which is perhaps the most widely used technique in the literature, has shown to be clearly inferior.

As a next step, the analysis should be extended to include arbitrary constraints, rather than only bounds on the range of values a variable is allowed to take on. We would expect, however, that the general insights gained from the study on bounds will be transferable to general constraint handling.

REFERENCES

[1] K.E. Parsopoulos and M.N. Vrahatis. Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, pages 235–306, 2002.

[2] Maurice Clerc. *Particle Swarm Optimization*. ISTE Ltd, 2006.

[3] James Kennedy and Russell C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann, 2001.

[4] Sabine Helwig and Rolf Wanka. Particle Swarm Optimization in High-Dimensional Bounded Search Spaces. In *Proceedings of the 2007 IEEE Swarm Intelligence Symposium*, pages 198–205, 2007.

TABLE VIII

AVERAGE DISTANCE (AND CORRESPONDING STANDARD ERROR) A PARTICLE MOVED PER ITERATION AND 100-DIMENSIONAL BENCHMARK. A REPRESENTATIVE SET OF BENCHMARKS WAS CHOSEN FOR THIS TABLE. THIS TABLE SHOWS THAT USING HYPERBOLIC BOUND HANDLING STRONGLY REDUCES THE AVERAGE PARTICLE SPEED WHILST SWARMS USING INFINITY BOUND HANDLING MOVE COMPARATIVELY FAST.

	Ackley	Rastrigin	Schwefel	f9	f10
Hyperbolic	0.67705±0.014152	1.4449±0.054519	237.99±7.1148	1.2394±0.06085	1.7454±0.028926
RandomBack	4.5511±0.033916	9.0959±0.15785	1278.6±23.228	7.1768±0.19236	7.2104±0.074256
Nearest-Z	3.9586±0.038494	7.3091±0.15584	993.23±18.361	5.4344±0.15655	5.4744±0.044543
Random-Z	3.4309±0.047696	6.3767±0.12325	855.16±12.243	4.791±0.1483	5.654±0.055334
Reflect-Z	3.7285±0.034692	6.4878±0.12871	671.66±25.59	4.6736±0.12349	5.0172±0.034842
BoundedMirror	8.1018±0.57238	7.8694±0.13905	1060.9±15.477	7.9261±0.18565	8.5677±0.095996
Infinity	138.91±0.96903	22.6±0.15665	1629.8±10.929	36.261±0.2083	36.46±0.22121
Infinity-C	26.679±0.43999	7.126±0.1007	805.55±9.3932	4.5751±0.16422	5.3285±0.051862

[5] C. Coello Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11):1245–1287, 2002.

[6] Z. Michalewicz. A survey of constraint handling techniques in evolutionary computation methods. In *4th Annual Conference on Evolutionary Programming*, pages 135–155, 1995.

[7] S. Janson and M. Middendorf. A hierachical particle swarm optimizer for dynamc optimization problems. In G. R. Raidl, editor, *Applications of Evolutionary Computing*, volume 3005 of *LNCS*, pages 513–524. Springer, 2004.

[8] J. Kennedy and R. Mendes. Population structure and particle swarm performance. In *Congress on Evolutionary Eomputation*, pages 1671–1676, 2002.

[9] Maurice Clerc and James Kennedy. The Particle Swarm - Explosion, Stability, and Convergence in a Multidimensional Complex Space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002.

[10] D. Bratton and J. Kennedy. Defining a standard for particle swarm optimization. In *Proceedings of the 2007 IEEE Swarm Intelligence Symposium*, 2007.

[11] Gregorio Toscano-Pulido and Carlos A. Coello Coello. A constraint-handling mechanism for particle swarm optimization. In *Proceedings CEC'04, the Congress on Evolutionary Computation*, pages 1396–1403, 2004.

[12] Sanaz Mostaghim, Werner E. Halter, and Anja Wille. Linear multi-objective particle swarm optimization. In Abraham, A.; Grosan, C.; Ramos, and V., editors, *Stigmergy optimization*, volume 31 of *Computational Science*, chapter 9, pages 209–237. Springer Verlag, JUL 2006.

[13] Maurice Clerc. Confinements and Biases in Particle Swarm Optimization. http://clerc.maurice.free.fr/pso/, 2006.

[14] J. E. Alvarez-Benitez, R. M. Everson, and J. E. Fieldsend. A MOPSO algorithm based exclusively on pareto dominance concepts. *Lecture Notes in Computer Science (LNCS), EMO 2005*, pages 459–473, 2005.

[15] Wen-Jun Zhang, Xiao-Feng Xie, and De-Chun Bi. Handling Boundary Constraints for Numerical Optimization by Particle Swarm Flying in Periodic Search Space. In *Proceedings of the IEEE Congress on Evol. Computation*, volume 2, pages 2307–2311, 2004.

[16] Daniel Bratton and James Kennedy. Defining a Standard for Particle Swarm Optimization. In *Proceedings of the 2007 IEEE Swarm Intelligence Symposium*, pages 120–127, 2007.

[17] Russell C. Eberhart and Yuhui Shi. Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization. In *Proceedings of the 2000 Congress on Evolutionary Computation*, pages 84–88, 2000.

[18] Bogdan Bochenek und Pawel Forys. Structural optimization against instability using particle swarms. In H. G. Beyer and et al., editors, *6th World Congress of Structural and Multidisciplinary Optimization*, 2005.

[19] Russell C. Eberhart and James Kennedy. A New Optimizer Using Particle Swarm Theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43, 1995.

[20] Rui Mendes. *Population Topologies and Their Influence in Particle Swarm Performance*. PhD thesis, Departamento de Informática, Escola de Engenharia, Universidade do Minho, 2004.

[21] James Kennedy and Rui Mendes. Population Structure and Particle Swarm Performance. In *Proceedings of the IEEE Congress on Evol. Computation*, pages 1671–1676, 2002.

[22] Alan Owen and Inman Harvey. Adapting particle swarm optimisation for fitness landscapes with neutrality. In *IEEE Swarm Intelligence Symposium*, pages 258–265. IEEE, 2007.

[23] Ponnuthurai N. Suganthan, Nikolaus Hansen, Jing J. Liang, Kalyanmoy Deb, Y.P. Chen, Anne Auger, and S. Tiwari. Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. KanGAL Report 2005005, Nanyang Technological University, Singapore, 2005.

[24] Sabine Helwig and Rolf Wanka. Theoretical Analysis of Initial Particle Swarm Behavior. In *Proceedings of the 10th International Conference on Parallel Problem Solving from Nature (PPSN08)*, pages 889–898, Dortmund, Germany, September 2008. Springer.