Particle Swarm Optimization with Velocity Adaptation

Sabine Helwig[*], Frank Neumann[†], and Rolf Wanka[*]
[*]Department of Computer Science, University of Erlangen-Nuremberg, Germany
{sabine.helwig, rwanka}@informatik.uni-erlangen.de
[†]Max-Planck-Institut für Informatik, Saarbrücken, Germany
fne@mpi-inf.mpg.de

Abstract—**Particle swarm optimization (PSO) algorithms have gained increasing interest for dealing with continuous optimization problems in recent years. Often such problems involve boundary constraints. In this case, one has to cope with the situation that particles may leave the feasible search space. To deal with such situations different bound handling methods have been proposed in the literature and it has been observed that the success of PSO algorithms depends on a large degree on the used bound handling method. In this paper, we propose an alternative approach to cope with bounded search spaces. The idea is to introduce a velocity adaptation mechanism into PSO algorithms that is similar to step size adaptation used in evolution strategies. Using this approach we show that the bound handling method becomes less important for PSO algorithms and that using velocity adaptation leads to better results for a wide range of benchmark functions.**

I. Introduction

Particle swarm optimization [8], [9] is a population-based stochastic search algorithm which has become very popular for solving continuous optimization problems in recent years. The algorithm is inspired by the interactions observed in social groups such as bird flocks, fish schools, or human societies. Continuous optimization problems are often defined on a bounded subspace of $\mathbb{R}^n$. Search points that do not belong to this area are considered as infeasible. One question is how to deal with particles that reach the infeasible region. Various methodologies were proposed in the literature, e.g., [3], [1]. Such methods often handle infeasible particles by guiding them towards the feasible region of the search space. Recently, it was observed that many particles leave the feasible region of the underlying search space at the beginning of the optimization process [7].

Our goal is to show how to avoid that many particles leave the feasible region of the parameter space of a bounded continuous optimization problem. We consider the model examined in [7] and point out that the problem of constructing many infeasible solutions occurs due to high velocities of the particles. Based on this observation it seems to be natural to place upper bounds on the velocities of the particles. To deal with the fact that sometimes high velocities are needed for an efficient search process, we propose to adjust the length of a particle's velocity vector in a similar way to step size adaptation in evolution strategies [11], [12]. This has the effect that if the algorithm is not able to construct feasible solutions

by high velocities, the velocities become smaller such that it is more likely to produce feasible solutions within the bounded search space. We examine the proposed velocity adaptation for particle swarm optimization and compare it to a standard particle swarm optimizer [2]. The effects of using different bound handling methods are investigated for the two approaches. When using no bounds or constant bounds on the velocities, the use of a suitable bound handling strategy is crucial for the success of the algorithm, especially for high-dimensional optimization problems. We show that our approach is almost invariant with respect to the bound handling method which means that we get more robust PSO algorithms by using velocity adaptation. Further experimental studies show that the use of velocity adaptation in PSO algorithms leads to significantly better results for a wide range of benchmark functions compared to a standard PSO approach.

The outline of the paper is as follows. In Section II, we describe particle swarm optimization for bounded search spaces. Section III considers the effect of velocities for staying within the boundary constraints. Based on these investigations, we introduce our novel particle swarm optimization algorithm using velocity adaptation in Section IV. Experimental investigations of our approach that point out the benefits of velocity adaption are reported in Section V. Finally, we finish with some concluding remarks.

II. Particle Swarm Optimization for Bound-Constrained Search Spaces

In this paper, we consider optimization problems with boundary constraints. This means that the search space $S = [lb_1, ub_1] \times [lb_2, ub_2] \times \ldots \times [lb_n, ub_n]$ consists of n real-valued parameters $(x_1, \ldots, x_n)$ where each parameter x_i, $1 \leq i \leq n$, is bounded with respect to some interval $[lb_i, ub_i]$. W.l.o.g., $S = [-r, r]^n$, i.e., $lb_i = -r$ and $ub_i = r$, $1 \leq i \leq n$.

Particle swarm optimization (PSO) [8], [9] is a population-based stochastic search algorithm for global optimization. Each individual, denoted as *particle*, moves through the n-dimensional search space S of an optimization problem with objective function $f : S \subseteq \mathbb{R}^n \to \mathbb{R}$. W.l.o.g., f is a minimization problem. Each particle i has a position $\vec{x}_{i,t}$, a velocity $\vec{v}_{i,t}$, a fitness value $f(\vec{x}_{i,t})$, and remembers the best position it has visited so far as its *private guide* $\vec{p}_{i,t}$. Furthermore, each particle i is able to communicate with a

subset of all particles, which is denoted as its *neighborhood*. In each iteration, it determines the best private guide of all its neighbors, its *local guide* $\vec{l}_{i,t}$. The iteration step is given by:

$$\vec{v}_{i,t} = \omega \cdot \vec{v}_{i,t-1} + c_1 \cdot \vec{r}_1 \odot (\vec{p}_{i,t-1} - \vec{x}_{i,t-1}) \qquad (1)$$
$$+ c_2 \cdot \vec{r}_2 \odot (\vec{l}_{i,t-1} - \vec{x}_{i,t-1})$$
$$\vec{x}_{i,t} = \vec{x}_{i,t-1} + \vec{v}_{i,t} \qquad (2)$$

where c_1, c_2, and ω are user-defined parameters, $\odot$ denotes component-wise vector multiplication, and $\vec{r}_1$ and $\vec{r}_2$ are vectors of random real numbers chosen uniformly at random in $[0, 1]$, and independently drawn every time they occur. After updating position and velocity of each particle, the private guides of successful particles are updated.

There exist a lot of strategies to handle boundary constraints in the literature, e.g., [3], [1], [14], [2], [7], among them:

- *Infinity:* Particles are allowed to move to an infeasible solution. In this case, its position and velocity remain unchanged, and the evaluation step is skipped [2].
- *Absorb:* Invalid particles are set on the nearest boundary position. The respective velocity components are set to zero [4].
- *Random:* Invalid components of a particle's position vector are set to a random value. Afterwards, the velocity is adjusted to $\vec{v}_{i,t+1} = \vec{x}_{i,t+1} - \vec{x}_{i,t}$ [14].

Previous studies analyzed the initial swarm behavior in bound-constrained search spaces [7]. For three different velocity strategies it was proven that many particles leave the search space in the first iteration with overwhelming probability w. r. t. the search space dimensionality. Hence, at least the initial particle swarm behavior strongly depends on the chosen bound handling mechanism. Experimental showed that modifying the bound handling strategy leads to significant performance differences on the investigated testproblems [7].

III. THEORETICAL OBSERVATIONS

In the following, we want to point out the effect of the velocity values when dealing with bounded search spaces. Our goal is to carry out a general investigation which is independent of the function that should be optimized. To examine the effect of maximum velocities we assume that the particles of a PSO algorithm are drawn uniformly at random from the underlying bounded search space. Additionally, we assume that the velocities are drawn from a fixed interval uniformly at random. Note that the given assumptions are often fulfilled in the first iteration of a PSO algorithm. As we do not take any properties of the optimization problem into account, we do not present rigorous results for later stages of the optimization process. Instead, we complement the theoretical studies carried out in this section with experimental investigations that confirm our findings. The following theorem generalizes the results given in [7] to velocities that are drawn uniformly at random from $[-\frac{r}{s}, \frac{r}{s}]^n$, where $s \geq 1$ is a parameter that determines the maximum velocity with respect to one particular direction.

Theorem 1: Let $\vec{x}_{i,t-1}$ and $\vec{v}_{i,t}$ be independently drawn from a uniform distribution over $[-r, r]^n$ and $[-\frac{r}{s}, \frac{r}{s}]^n$ with $s \geq 1$, respectively, for an arbitrary particle i at time step t. Then, the probability that particle i leaves the search space in iteration t is $1 - \left(1 - \frac{1}{4s}\right)^n$.

Proof: According to Eq. (2), particle i's position at time step t evaluates to $\vec{x}_{i,t} = \vec{x}_{i,t-1} + \vec{v}_{i,t}$ Hence, the d-th component of $\vec{x}_{i,t}$, $x_{i,t,d}$, is the sum of two independent uniformly distributed random variables. The density function $f_{x_{i,t,d}}$ of $x_{i,t,d}$ is trapezoidal and can be determined by convolution to

$$f_{x_{i,t,d}}(z) = \begin{cases} \frac{s}{4r^2}z + \frac{s}{4r} + \frac{1}{4r} & \text{for } -r - \frac{r}{s} \leq z \leq \frac{r}{s} - r \\ \frac{1}{2r} & \text{for } -r + \frac{r}{s} < z < r - \frac{r}{s} \\ -\frac{s}{4r^2}z + \frac{s}{4r} + \frac{1}{4r} & \text{for } -\frac{r}{s} + r \leq z \leq r + \frac{r}{s} \\ 0 & \text{otherwise .} \end{cases}$$

Thus, the probability that particle i leaves the search space in dimension d is

$$\mathrm{Prob}(x_{i,t,d} < -r) + \mathrm{Prob}(x_{i,t,d} > r)$$
$$= \int_{-\infty}^{-r} f_{x_{i,t,d}}(z)\mathrm{d}z + \int_{r}^{\infty} f_{x_{i,t,d}}(z)\mathrm{d}z = \frac{1}{4s} .$$

As each component of $\vec{x}_{i,t}$ is computed independently, the probability $p(n, s)$ that particle i becomes infeasible in iteration t evaluates to

$$p(n, s) = \mathrm{Prob}(\vec{x}_{i,t} \notin [-r, r]^n) = 1 - \left(1 - \frac{1}{4s}\right)^n .$$

∎

Note, that $\lim_{n \to \infty}(1 - \left(1 - \frac{1}{4s}\right)^n) = 1 - e^{-\frac{n}{4s}}$ holds. Theorem 1, shows that the probability that a particle leaves the bounded search space crucially depends on the interval from which the velocities are chosen. For example, if s is a constant, the probability that a particle becomes infeasible rapidly approaches 1 when the dimensionality n of the optimization problem increases. On the other hand, $s = n$ implies that, with constant probability, particles do not leave the bounded search space.

This theoretical result stresses the importance of adapting velocities during the optimization process. Large velocities are important for search space exploration and to gain large improvements. However, the probability that particles leave the search space increases when velocities are allowed to grow large, which might deteriorate the swarm's performance. Reducing the particles' velocities also reduces the probability that particles become infeasible, down to a constant probability if velocities are, for instance, drawn uniformly at random in $[-\frac{r}{n}, \frac{r}{n}]^n$. Hence, small velocities are necessary to cope with situations where large improvements are not possible. These considerations lead us to the PSO with velocity adaptation presented in the subsequent section.

IV. PSO WITH VELOCITY ADAPTATION

The investigations carried out in the previous section have pointed out the influence of velocities on the probability that particles leave the bounded search space. It has been shown how the probability of leaving the bounded search space

depends on the interval from which the velocity is chosen. The probability of leaving the bounded search space decreases when the velocities decrease. However, small velocities lead to stagnation which implies that the algorithm achieves less progress. Therefore, we propose to use velocity adaptation in particle swarm optimization such that a large progress can be achieved by high velocities. On the other hand, low velocities should be attained if the algorithm observes to have difficulties for finding better solutions. Our scheme for velocity adaptation is similar to the use of step size adaptation known in the field of evolution strategies. The idea is to increase velocities in the case that the particles have found improvements during the last iterations. In the case that no progress could be achieved, we reduce the velocities. The amount of progress that has been made during the last iterations is measured by the number of successes. A success occurred if the private guide of a particle is updated to its current position. This happens if the new particle position is strictly better than its private guide. If the new particle position and the private guide are of equal fitness, the private guide is updated with probability $1/2$.

Definition 1 (Success): A particle i is called *successful* in iteration t if its private guide $\vec{p}_{i,t}$ is updated to its current position $\vec{x}_{i,t}$.

Our particle swarm optimizer using velocity adaptation is shown in Algorithm 1. It distinguishes itself from standard PSO approaches by using the described velocity adaptation mechanism. The algorithm uses a counter $SuccessCounter$ which counts the number of successes according to Definition 1. In total n iterations are considered to decide whether velocities are increased or decreased. The success rate of the algorithm for a number of n iterations is given by $SuccessCounter/n$. If the observed success rate is higher than a given threshold (called $SuccessProbability$) the velocities are increased by a factor of 2 otherwise the velocities are scaled down by a factor of $1/2$. In most of our experimental studies we will use a success probability of 0.2 which is motivated by the $1/5$-rule used in evolution strategies [11].

V. Experimental Results

In the subsequent experimental analysis, our novel PSO with velocity adaptation is compared to a standard particle swarm optimizer [2]. For both approaches, the same parameter setting was used: $c_1 = c_2 = 1.496172$, $\omega = 0.72984$. As proposed by Kennedy and Mendes, the swarm is connected via the so-called *von Neumann* topology, a two-dimensional grid with wrap-around edges [10]. The population size was set to $m = 49$ to arrange the particles in a 7×7 grid. In the non-adaptive PSO, velocity clamping was applied, i.e., each velocity component is restricted to $[-r \dots r]$ [5], where $[-r \dots r]^n$ is the parameter space of the optimization problem to be solved. For the PSO with velocity adaptation, a success probability of 0.2 was used, and the initial velocity length was set to r. Note that Experiment 2 and 3 study the effect of these parameters.

Experiment 1 compares our new algorithm to a standard PSO. Therefore, six widely-used benchmarks, Sphere, Rosenbrock, Rastrigin, Griewank, Schwefel 2.6, and Ackley (func-

---

Algorithm 1 PSO with velocity adaptation
Require: Objective function $f : \mathcal{S} \subseteq \mathbb{R}^n \to \mathbb{R}$, $SuccessProbability \in R_{[0,1]}$, Initial velocity length l
1: Initialize particle positions, velocities, and private guides
2: Initialize neighborhood graph
3: $SuccessCounter \leftarrow 0$
4: $VelocityLength \leftarrow l$
5: Scale each initial velocity (up or down) so that its length is exactly $VelocityLength$
6: **for** $t := 1, \dots, maxIteration$ **do**
7:     **for** $i := 0, \dots, m - 1$ **do**
8:         Velocity update according to Eq. (1)
9:         Scale velocity (up or down) so that its length is exactly $VelocityLength$
10:         Position update according to Eq. (2)
11:     **end for**
12:     **for** $i := 0, \dots, m - 1$ **do**
13:         Update private guide of particle i
14:         **if** (success($\vec{x}_{i,t}, \vec{p}_{i,t-1}$)) **then**
15:             $SuccessCounter \leftarrow SuccessCounter + 1$
16:         **end if**
17:     **end for**
18:     **if** $t \mod n = 0$ **then**
19:         $SuccessRate \leftarrow \frac{SuccessCounter}{n}$
20:         **if** $SuccessRate > SuccessProbability$ **then**
21:             $VelocityLength \leftarrow 2 \cdot VelocityLength$
22:         **else**
23:             $VelocityLength \leftarrow \frac{VelocityLength}{2}$
24:         **end if**
25:         $SuccessCounter \leftarrow 0$
26:     **end if**
27: **end for**

---

tion descriptions see, e.g., [2]), and the CEC 2005 benchmarks f1–f14 [13], were used as 100- and 500-dimensional problems. From the CEC 2005 benchmarks, all noisy functions and all functions without search space boundaries were excluded. Additionally, f8 was omitted because all swarms were equally unsuccessful in preliminary experiments, and f12 was skipped due to its high computation time. For all benchmarks, particles were initialized uniformly at random in the whole search space. Historically, individuals are often initialized in a subspace of the feasible region in order to avoid that the performance of algorithms with center bias is overestimated. E.g., the initialization space of the Sphere function may be set to $[50 \dots 100]^n$ whereas the parameter space is $[-100 \dots 100]^n$ [2]. However, we do not use these asymmetric initialization ranges due to the following two reasons: First, we also investigate several CEC 2005 benchmarks which do not have their global optimum at the center of the search space. Hence, center bias can be identified considering all results. Second, we explicitly do not want to design an algorithm which has to explore outside the initialization space as such behavior is not needed for real applications. Velocities were

initialized with half-diff initialization [7]. Experiment 2 and 3 are dedicated to the investigation of different parameter settings for our new PSO algorithm, and were only run on 100-dimensional problems.

In our experimental study, each run was terminated after $300,000$ function evaluations, and each configuration was repeated 50 times. The performance of two algorithms A and B is compared by using the one-sided Wilcoxon rank sum test with null-hypothesis $H_0 : F_A(z) = F_B(z)$, and the one-sided alternative $H_1 : F_A(z) < F_B(z)$, where $F_X(z)$ is the distribution of the results of algorithm X. The significance level was set to $\alpha = 0.01$. Additionally, we show the obtained average objective values and corresponding standard errors as well as convergence plots for selected functions where appropriate.

A. Experiment 1: Comparison with a Standard PSO

The novel PSO with velocity adaptation was compared to a standard particle swarm optimizer on all 100- and 500-dimensional problems. The obtained average values and corresponding standard errors are shown in Table I. The results of the one-sided Wilcoxon rank sum tests are shown in Table II. The experiment was conducted to investigate the following two questions: First, which algorithm, the adaptive PSO or the standard PSO, performs better on the chosen testbed? Second, is the new algorithm more invariant with respect to the bound handling mechanism than a standard particle swarm optimizer?

Table I shows that the adaptive PSO (denoted as *Absorb-A*, *Random-A*, and *Infinity-A*, depending on the bound handling method) provides superior average results than the used standard PSO (*Absorb-S*, *Random-S*, *Infinity-S*) on most testfunctions. The only exceptions for the 100-dimensional functions are Schwefel, f9, and f10. When solving the 500-dimensional test suite, exceptions are again f9 and f10, but the relative difference in the average objective values is diminished. In some 500-dimensional cases, *Random-A* was not able to provide satisfactory results for the optimization problem under consideration. It was observed earlier that *Random* can distract particles from the boundary [6] which might be a reason for the bad performance of *Random-A* on some functions. Note that in high-dimensional spaces, most of the volume is located near the boundary (see, e.g., Theorem 3.1. in [6]), and a search algorithm should therefore be able to explore boundary regions. Table II lists the results of the one-sided Wilcoxon rank sum test applied on the outcome of this experiment. The tables show that the adaptive variants significantly outperform their non-adaptive counterparts on much more testfunctions (highlighted by bold numbers) than vice versa (italic numbers).

The second question was whether the adaptive PSO is more invariant to the bound handling strategy compared to the standard PSO. We already mentioned an exception: *Random-A* was sometimes significantly outperformed by *Absorb-A* and *Infinity-A*, especially in the 500-dimensional case. Hence, our adaptive mechanism sometimes cannot prevent that particles are not able to converge to boundary regions when using *Random* bound handling. However, a comparison of the average

objective values presented in Table I shows that invariance with respect to the bound handling strategy was nevertheless improved: The obtained avarage objective values of the three adaptive variants are mostly more similar to one another than the ones of the non-adaptive PSO. This observation is confirmed by the results of the Wilcoxon rank sum test presented in Table II: Especially in the 500-dimensional case, *Absorb-S*, *Random-S*, and *Infinity-S* showed significant performance differences among each other. *Infinity-S* was outperformed by the other two variants in 15 of 16 functions. Although performance differences can be observed among the adaptive PSOs as well, they occur less often. *Absorb-A* and *Infinity-A* provided solutions of similar quality on most 500-dimensional problems. We conclude that we partly achieved the second goal, but there is still room for improvement. Experiment 3 will show that initializing the velocity length to a smaller value can enhance invariance with respect to the bound handling strategy but also deteriorate performance.

Summarized, PSO with velocity adaptation yielded superior results than a standard PSO on most of the investigated 100- and 500-dimensional benchmarks. Invariance with respect to the bound handling mechanism was enhanced but is still not perfect.

B. Experiment 2: Success probability

The adaptation mechanism of our adaptive PSO is similar to the well-known *1/5 rule of Rechenberg* [11]: Whenever the success rate in the past n iterations exceeds $1/5$, the velocity is doubled, otherwise it is halved. Of course, other values besides $1/5$ can be chosen as success probablity threshold. We additionally investigated the following settings: 0.01, 0.1, 0.5, and 0.8. In Figure 1, two representative runs are shown. Setting the success probability to 0.5 or 0.8 performed bad on most benchmarks, and can therefore not be recommended. The performance of 0.1 and 0.2 was quite similar with slight advantage for 0.2 considering average objective values and the result of the Wilcoxon rank sum tests for all functions. The behavior of choosing 0.01 differs from the other settings: Convergence is very slow, often too slow to obtain satisfactory results using $300,000$ function evaluations (which already is quite much). When applying such a low success probability, we expect velocities to grow large. This can lead to a rather random behavior of the whole swarm and prevent convergence, as shown in Figure 1 (top) and observed on approximately 7 functions. On the other hand, the same behavior is advantageous for other problems, see Figure 1 (bottom): In these cases, using high success probabilities led to premature convergence, as velocities are decreased further and further if the swarm is not successful. However, the swarms using a threshold success probability of 0.01 keep exploring and finally found solutions of much better quality on approximately 6 of the investigated 16 benchmark functions. Summarized, both very low and very high success probability thresholds cannot be recommended. Using a very low success probability sometimes leads to exceptionally good results, but also often deteriorates particle swarm performance. However,

100-dimensional benchmarks

	Sphere (0)	Rosenbrock (0)	Ackley (0)	Griewank (0)	Rastrigin (0)	Schwefel ($\approx$ -41898.3)
Absorb-S	6.0693e-06$\pm$1.175e-07	191.06$\pm$8.785	1.3959$\pm$0.11837	2.765e-03$\pm$7.7612e-04	282.2$\pm$4.504	-27841$\pm$242.12
Random-S	6.0783e-06$\pm$1.2964e-07	195.45$\pm$8.2053	1.7332$\pm$0.10218	3.7511e-03$\pm$7.7485e-04	239.56$\pm$4.6447	-24826$\pm$207.29
Infinity-S	6.2083e-06$\pm$1.4284e-07	221.06$\pm$7.0927	1.5847$\pm$0.1094	7.2517e-03$\pm$2.5285e-03	276.34$\pm$5.6791	-23705$\pm$234.82
Absorb-A	1.0473e-06$\pm$9.3267e-09	114.03$\pm$4.7795	3.7094e-06$\pm$1.3119e-08	2.7088e-03$\pm$8.7574e-04	93.91$\pm$2.3929	-24430$\pm$180.2
Random-A	1.0589e-06$\pm$1.0115e-08	120.75$\pm$4.5423	3.6963e-06$\pm$1.5878e-08	1.4789e-03$\pm$6.1713e-04	87.716$\pm$2.1884	-22341$\pm$170.7
Infinity-A	1.0437e-06$\pm$9.9384e-09	107.08$\pm$3.6154	3.7032e-06$\pm$1.7861e-08	5.9275e-04$\pm$3.4177e-04	93.499$\pm$2.3445	-22837$\pm$187.56

	f1 (-450)	f2 (-450)	f3 (-450)	f5 (-310)	f6 (390)	f9 (-330)
Absorb-S	-450$\pm$0	25223$\pm$1166.3	39163000$\pm$1884400	27545$\pm$501.09	594.24$\pm$8.27	29.387$\pm$8.8699
Random-S	-450$\pm$0	23035$\pm$973.35	78163000$\pm$5844400	30534$\pm$497.03	590.71$\pm$8.142	48.163$\pm$6.6587
Infinity-S	-450$\pm$0	9951.7$\pm$395.9	17457000$\pm$727450	33148$\pm$485.31	593.54$\pm$6.4731	226.72$\pm$12.379
Absorb-A	-450$\pm$0	286.74$\pm$27.998	8120300$\pm$222840	23100$\pm$333.67	511.6$\pm$5.3632	147.44$\pm$10.078
Random-A	-450$\pm$0	1685.6$\pm$733.36	1.1002e+08$\pm$12771000	35603$\pm$639.93	530.09$\pm$6.3345	160.73$\pm$6.9788
Infinity-A	-450$\pm$0	269.45$\pm$33.939	7234600$\pm$154480	26836$\pm$355.33	517.21$\pm$5.576	174.25$\pm$10.714

	f10 (-330)	f11 (-460)	f13 (-130)	f14 (-300)		
Absorb-S	30.313$\pm$7.7389	217.56$\pm$0.81253	-69.427$\pm$1.9555	-253.47$\pm$0.058467		
Random-S	98.299$\pm$8.9825	221.94$\pm$0.88144	-68.36$\pm$1.5856	-253.7$\pm$0.068529		
Infinity-S	261.59$\pm$13.889	224.14$\pm$0.94733	-69.666$\pm$1.6095	-253.54$\pm$0.053858		
Absorb-A	85.767$\pm$9.1288	185.36$\pm$1.1865	-112.95$\pm$0.45879	-254.01$\pm$0.070051		
Random-A	149.52$\pm$10.355	195.1$\pm$1.2822	-113.28$\pm$0.36477	-254.13$\pm$0.060984		
Infinity-A	195.87$\pm$10.924	185.43$\pm$1.2738	-113.12$\pm$0.48073	-254.15$\pm$0.085889		

500-dimensional benchmarks

	Sphere (0)	Rosenbrock (0)	Ackley (0)	Griewank (0)	Rastrigin (0)	Schwefel ($\approx$ -209491.5)
Absorb-S	1669.3$\pm$236.6	397590$\pm$32299	9.1571$\pm$0.13512	15.692$\pm$1.9949	1917.1$\pm$29.846	-122640$\pm$847.19
Random-S	1523.8$\pm$136.19	877140$\pm$157910	9.4522$\pm$0.16661	15.428$\pm$1.1851	1642$\pm$25.489	-106940$\pm$647.94
Infinity-S	780890$\pm$5098.7	2.3589e+09$\pm$22756000	20.057$\pm$0.018442	7023.7$\pm$47.232	6704.4$\pm$21.475	-11628$\pm$276.33
Absorb-A	0.54291$\pm$0.15378	2188$\pm$45.632	1.712$\pm$0.030091	0.19666$\pm$0.050727	422.85$\pm$13.534	-99575$\pm$981.1
Random-A	0.8328$\pm$0.35029	2232.3$\pm$53.622	1.7225$\pm$0.028023	0.12122$\pm$0.029484	375.63$\pm$10.86	-88646$\pm$985.12
Infinity-A	0.4738$\pm$0.095705	2210.7$\pm$36.524	1.762$\pm$0.025693	0.09308$\pm$0.017517	408.62$\pm$11.456	-93069$\pm$973.41

	f1 (-450)	f2 (-450)	f3 (-450)	f5 (-310)	f6 (390)	f9 (-330)
Absorb-S	3260.3$\pm$296.88	2369600$\pm$32679	1.6426e+09$\pm$28943000	-310$\pm$0	6.1811e+08$\pm$86998000	3288.4$\pm$40.28
Random-S	31185$\pm$1302.3	2556700$\pm$40344	3.7348e+09$\pm$1.2208e+08	-310$\pm$0	1.5943e+09$\pm$1.5081e+08	3218.5$\pm$29.644
Infinity-S	2174600$\pm$7890.1	97764000$\pm$5238200	1.1268e+11$\pm$1.2292e+09	-310$\pm$0	2.0502e+12$\pm$1.4785e+10	10220$\pm$23.611
Absorb-A	-430.82$\pm$4.9622	1071200$\pm$18936	4.795e+08$\pm$5962900	-310$\pm$0	705350$\pm$83435	3306.4$\pm$18.958
Random-A	201560$\pm$4265.1	2999200$\pm$123610	6.3402e+09$\pm$1.2971e+08	-310$\pm$0	4.5585e+09$\pm$4.7332e+08	3417$\pm$18.45
Infinity-A	-414.15$\pm$20.668	866340$\pm$10705	4.2855e+08$\pm$6008000	-310$\pm$0	309230$\pm$58497	3352.9$\pm$20.115

	f10 (-330)	f11 (-460)	f13 (-130)	f14 (-300)		
Absorb-S	4959.3$\pm$74.314	909.67$\pm$2.2174	10650$\pm$1854.9	-54.763$\pm$0.050616		
Random-S	5560.7$\pm$74.102	918.3$\pm$1.844	3594.6$\pm$227.73	-54.835$\pm$0.062736		
Infinity-S	16972$\pm$48.023	1025.1$\pm$1.5096	45526000$\pm$1002400	-52.774$\pm$0.05418		
Absorb-A	7881.6$\pm$46.142	751$\pm$3.701	404.46$\pm$10.313	-60.306$\pm$0.1963		
Random-A	7975.7$\pm$64.224	849.35$\pm$5.8459	409.14$\pm$11.209	-59.999$\pm$0.15094		
Infinity-A	7803.3$\pm$39.094	756.8$\pm$3.822	406.86$\pm$11.149	-60.105$\pm$0.14006		

values of 0.1 or 0.2 delivered solutions of good quality on most investigated problems.

C. Experiment 3: Initialization of velocity length l

The initial velocity length determines the exploration behavior of the particle swarm at the beginning of the optimization process. Choosing very small initial velocities can result in premature convergence on local optima. Two different settings were tested: $l = r$ and $l = r/\sqrt{n}$, where $[-r \dots r]^n$ is the parameter space of the problem to be solved. The second setting is denoted as *init2*. Comparing the obtained average objective values (not presented here), initializing the velocity length to $l = r$ provided better results on most benchmark functions. In the other cases, results of similar quality were achieved. Especially for Schwefel, f9 and f10, great performance losses could be recognized when initializing the velocity length to $l = r/\sqrt{n}$. However, as indicated by the theoretical study in Section III, choosing $l = r/\sqrt{n}$ (which corresponds to the order r/n per dimension) results in a PSO algorithm which is more invariant with respect to the bound handling method:

Table II (right) shows that *Absorb-init2*, *Random-init2*, and *Infinity-init2* less often significantly differed in performance among each other than *Absorb-A*, *Random-A*, and *Infinity-A*. However, the difference is rather small so that initializing velocities to $l = r$ is to be preferred due to the desired explorative behavior of a search algorithm at the beginning of the optimization process. Considering Table II (right), it is clear that the *Absorb-A*, *Random-A*, and *Infinity-A* variants more often significantly outperform the corresponding *init2* algorithms than vice versa. In the three cases in which the *init2* variants performed significantly better than the adaptive PSO with $l = r$ (Sphere, Ackley, and Griewank), both settings provided a satisfactory average solution quality very close to the global optimum.

VI. CONCLUSION

Particle swarm optimization algorithms have found many applications in solving continuous optimization problems. Often such problems have boundary constraints and it has been observed that PSO algorithms have difficulties when dealing

TABLE II
SUMMARY OF ONE-SIDED WILCOXON RANK SUM TEST WITH SIGNIFICANCE LEVEL 0.01 FOR EXPERIMENT 1 AND 3. FOR EACH ALGORITHMIC COMBINATION (A, B), THIS MATRIX SHOWS HOW OFTEN A SIGNIFICANTLY OUTPERFORMED B. *Example:* ENTRY 6 IN THE FIRST ROW OF THE FIRST TABLE SHOWS THAT ABSORB-S SIGNIFICANTLY OUTPERFORMED INFINITY-S ON 6 BENCHMARKS.

Experiment 1: 100D

	1	2	3	4	5	6
Absorb-S (1)	0	5	6	*3*	5	3
Random-S (2)	1	0	5	1	*4*	3
Infinity-S (3)	2	2	0	0	3	*1*
Absorb-A (4)	**12**	12	14	0	7	3
Random-A (5)	10	**10**	12	0	0	1
Infinity-A (6)	11	12	**14**	1	4	0
Total number of benchmarks: 16						

Experiment 1: 500D

	1	2	3	4	5	6
Absorb-S (1)	0	8	15	*2*	7	3
Random-S (2)	2	0	15	3	*6*	3
Infinity-S (3)	0	0	0	0	0	*0*
Absorb-A (4)	**12**	12	15	0	7	1
Random-A (5)	8	**8**	15	1	0	1
Infinity-A (6)	12	12	**15**	3	7	0
Total number of benchmarks: 16						

Experiment 3

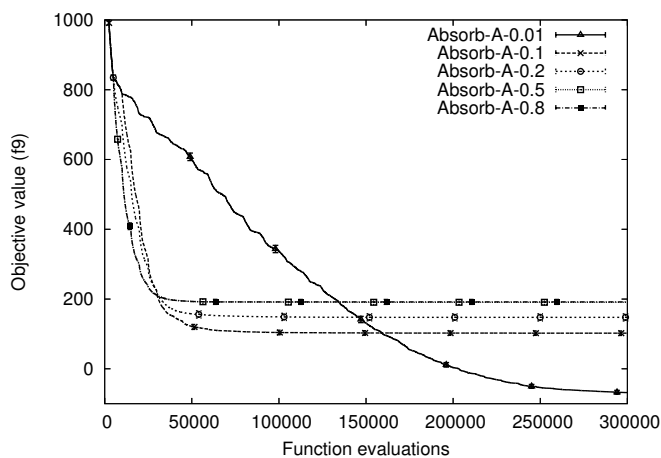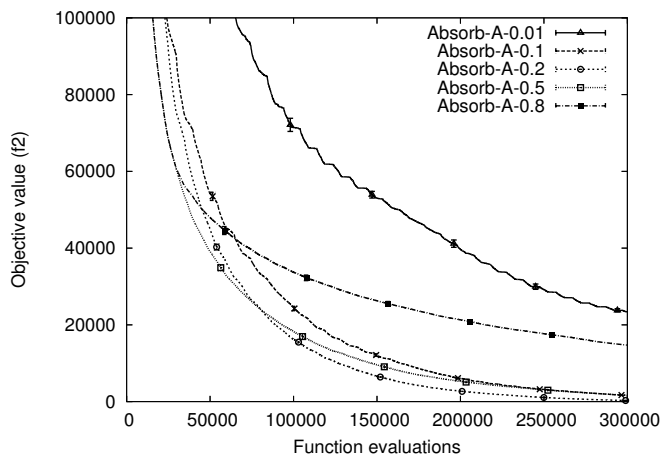|	1	2	3	4	5	6
Absorb-A (1)	0	7	3	**8**	9	7
Random-A (2)	0	0	1	6	**9**	6
Infinity-A (3)	1	4	0	8	9	**9**
Absorb-init2 (4)	*3*	6	3	0	4	0
Random-init2 (5)	3	*3*	3	0	0	0
Infinity-init2 (6)	3	6	*3*	0	3	0
Total number of benchmarks: 16						

Fig. 1. Representative runs (top: f2, bottom: f9) of the adaptive PSO with different success probabilities (Experiment 2). In the plot, average objective values are shown. Vertical bars (very small) depict the standard error.

with such problems as many particles leave the feasible region of the search space due to high velocities. Because of this, different bound handling methods have been proposed in the literature and it has been observed that these methods have a large impact on the success of particle swarm optimization algorithms.

Our theoretical investigations explain the effect of velocity values from a theoretical point of view. Based on these obser-vations, we proposed a particle swarm optimization algorithm which uses velocity adaptation to deal with bound-constrained search spaces. This novel algorithm deals with boundary constraints by decreasing the velocities when no progress can be achieved. Our experimental analysis shows that the influence of the used bound handling method becomes less important when using velocity adaptation. Furthermore, the use of velocity adaptation leads to significantly better results for many benchmark functions in comparison to a standard particle swarm optimizer.

REFERENCES

[1] Julio E. Alvarez-Benitez, Richard M. Everson, and Jonathan E. Field-send. A MOPSO Algorithm Based Exclusively on Pareto Dominance Concepts. In *Evolutionlary Multi-Criterion Optimization*, pages 459–473, 2005.
[2] Daniel Bratton and James Kennedy. Defining a Standard for Particle Swarm Optimization. In *Proc. IEEE Swarm Intelligence Symp.*, pages 120–127, 2007.
[3] Maurice Clerc. Confinements and Biases in Particle Swarm Optimization. http://clerc.maurice.free.fr/pso/, 2006.
[4] Maurice Clerc. *Particle Swarm Optimization*. ISTE Ltd, 2006.
[5] 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.
[6] Sabine Helwig and Rolf Wanka. Particle Swarm Optimization in High-Dimensional Bounded Search Spaces. In *Proc. IEEE Swarm Intelligence Symp.*, pages 198–205, 2007.
[7] Sabine Helwig and Rolf Wanka. Theoretical Analysis of Initial Particle Swarm Behavior. In *Proc. of the 10th Int. Conference on Parallel Problem Solving from Nature (PPSN08)*, pages 889–898. Springer, 2008.
[8] James Kennedy and Russell C. Eberhart. Particle Swarm Optimization. In *Proc. of the IEEE Int. Conf. on Neural Networks*, volume 4, pages 1942–1948, 1995.
[9] James Kennedy and Russell C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann, 2001.
[10] James Kennedy and Rui Mendes. Population Structure and Particle Swarm Performance. In *Proceedings of the IEEE Congress on Evol. Computation*, pages 1671–1676, 2002.
[11] Ingo Rechenberg. *Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog Verlag, 1973.
[12] Hans-Paul Paul Schwefel. *Evolution and Optimum Seeking: The Sixth Generation*. John Wiley & Sons, Inc., New York, NY, USA, 1993.
[13] 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.
[14] 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 *Proc. of the IEEE Congress on Evol. Computation*, volume 2, pages 2307–2311, 2004.