Chapter 1

PARTICLE SWARM OPTIMIZATION AND SEQUENTIAL SAMPLING IN NOISY ENVIRONMENTS

Thomas Bartz-Beielstein
Chair of Algorithm Engineering and Systems Analysis,
Department of Computer Science, Dortmund University, Germany
thomas.bartz-beielstein@udo.edu

Daniel Blum
Chair of Algorithm Engineering and Systems Analysis,
Department of Computer Science, Dortmund University, Germany
daniel.blum@udo.edu

Jürgen Branke
Institute AIFB, University of Karlsruhe
76128 Karlsruhe, Germany
branke@aifb.uni-karlsruhe.de

Abstract For many practical optimization problems, the evaluation of a solution is subject to noise, and optimization heuristics capable of handling such noise are needed. In this paper, we examine the influence of noise on particle swarm optimization and demonstrate that the resulting stagnation can not be removed by parameter optimization alone, but requires a reduction of noise through averaging over multiple samples. In order to reduce the number of required samples, we propose a combination of particle swarm optimization and a statistical sequential selection procedure, called optimal computing budget allocation, which attempts to distribute a given number of samples in the most effective way. Experimental results show that this new algorithm indeed outperforms the other alternatives.

Keywords: Particle Swarm Optimization, Noise, Sequential Sampling

Introduction

In many real-world optimization problems, solution qualities can only be estimated but not determined precisely. Falsely calibrated measurement instruments, inexact scales, scale reading errors, etc. are typical sources for measurement errors. If the function of interest is the output from stochastic simulations, then the measurements may be exact, but some of the model output variables are random variables. The term "noise" will be used in the remainder of this article to subsume these phenomena.

This article discusses the performance of *particle swarm optimization* (PSO) algorithms on functions disturbed by Gaussian noise. It extends previous analysis by also examining the influence of algorithm parameters, by considering a wider spectrum of noise levels, and analyzing different types of noise (multiplicative and additive). Furthermore, we integrated a recently developed sequential sampling technique into the particle swarm optimization method. Similar techniques have been integrated into other metaheuristics like evolutionary algorithms, but their application to the PSO algorithm is new.

The paper is structured as follows. First, we briefly introduce PSO in Section 1. Then, the effects of noise and sequential sampling techniques are discussed in Section 2. Section 4 presents several experimental results, including the effect of parameter tuning, some algorithmic variants with perfect local and global knowledge, and the integration of sequential sampling. The paper concludes with a summary and an outlook.

1. Particle swarm optimization

PSO uses a population (*swarm*) of *particles* to explore the search space. Each particle represents a candidate solution of an optimization problem and has an associated position, velocity, and memory vector. The main part of the PSO algorithm can be described formally as follows: Let $S \subseteq \mathbb{R}^n$ be the n-dimensional search space of a (fitness) function $f : S \to \mathbb{R}$ to be optimized. Without loss of generality, throughout the rest of this article, optimization problems will be formulated as minimization problems. Assume a swarm of s particles. The ith particle consists of three components. The first one, x_i, is its position in the search space, the second component, v_i, describes the velocity, and the third component, p_i^*, is its memory, storing the best position encountered so far. This vector is often referred to as *personal best* in the PSO literature. Finally, the term p_g^* denotes the best position position found so far by the whole swarm, and is generally referred to as *global best*. Let t denote the current generation. Velocities and positions are updated for each

dimension $1 \leq d \leq n$ as follows:

$$v_i(t+1) = \underbrace{w v_i(t)}_{\text{momentum}} + \underbrace{c_1 u_{1i} \left\{p_i^*(t) - x_i(t)\right\}}_{\text{local information}} + \underbrace{c_2 u_{2i} \left\{p_g^*(t) - x_i(t)\right\}}_{\text{global information}}, \tag{1.1}$$

$$x_i(t+1) = x_i(t) + v_i(t+1).$$

Before optimization can be started, several parameters or factors have to be specified. These so-called *exogenous* factors will be analyzed in more detail below. Parameters that are used inside the algorithm are referred to as *endogenous*. The endogenous factors u_{1i} and u_{2i} are realizations of uniformly distributed random variables U_{1i} and U_{2i} in the range $[0, 1]$. The exogenous factors c_1 and c_2 are weights that regulate the influence of the local and the global information. The factor w in the *momentum* term of Eq. 1.1 is called *inertia weight*. It was introduced in Shi and Eberhart (1998) to balance the global and local search abilities.

2. The effect of noise

Noise makes it difficult to compare different solutions and select the better ones. In PSO, noise affects two operations: In every iteration,

1 each particle has to compare the new solution to its previous best and retain the better one, and

2 the overall best solution found so far has to be determined.

Wrong decisions can cause a *stagnation* of the search process: Overvaluated candidates—solutions that are only seemingly better—build a barrier around the optimum and prevent convergence. The function value at this barrier will be referred to as the *stagnation level*. Or, even worse, the search process can be *misguided:* The selection of seemingly good candidates moves the search away from the optimum. This phenomenon occurs if the noise level is high and the probability of a correct selection is very small.

There is very little research on how strongly the noise affects the overall performance of PSO, and what measures are suitable to make PSO more robust against noise. Parsopoulos and Vrahatis (2001) were probably the first to present some results regarding the behavior of the PSO algorithm in noisy and continuously changing environments. In Parsopoulos and Vrahatis (2002), they focused on noise alone and concluded that "... in the presence of noise the PSO method is very stable and efficient." In both papers, fitness proportional noise models were used.

Krink et al. (2004) compared differential evolution, evolutionary algorithms, and PSO on noisy fitness functions. The noise was independent of the solution's fitness. To reduce the effect of the noise, they suggested to average over a number of samples. Finally, in Liu et al. (2005), PSO is combined with local simulated annealing and a hypothesis test to tackle flow shop problems with noisy processing times. Thereby, the hypothesis tests are used to decide whether the new solution should replace a particle's personal best position.

The influence of noise on *evolutionary algorithms* (EAs) has received much more attention. EAs have been shown to work quite well in the presence of noise. Also, it has been proven analytically that under certain conditions, increasing the population size may help an evolution strategy to cope even better with the noise (Beyer, 2001). Several papers report on the successful integration of statistical tests or selection procedures into evolutionary algorithms, see, e.g., Rudolph (1997); Bartz-Beielstein and Markon (2004); Branke and Schmidt (2004); Buchholz and Thümmler (2005); Schmidt et al. (2006). A comprehensive overview on the topic of evolutionary algorithms in the presence of noise is given in Jin and Branke (2005). Based on the good performance of evolution strategies in noisy environments, one might hope that also PSO can somehow cope with the noise, and that it is sufficient to adjust its parameters.

Alternatively, one may attempt to reduce the effect of noise explicitly. The simplest way to do so is to sample a solution's function value n times, and use the average as estimate for the true expected function value. While this reduces the standard deviation of the mean by a factor of $\sqrt{n}$, it also increases the running time by a factor of n, which is often not acceptable.

3. Optimal computing budget allocation

We consider statistical selection procedures that use only a small number of samples to identify the best out of a set of solutions with a high probability. There is a multitude of selection procedures in the literature. Bechhofer et al. (1995) give a comprehensive introduction into statistical selection methods. *Two-stage procedures* use the samples of a first stage to estimate means and variances. In the second stage, an additional amount of samples is drawn for each candidate solution, each amount depending on the variance and the overall required probability of correct selection. *Sequential procedures* allow even more than two stages. Such methods use either an elimination mechanism to reduce the number of alternatives considered for sampling, or they assign additional

samples only to the most promising alternatives. The intuition is to use all available information as soon as possible to adjust the further process in a promising way. The most sophisticated sampling approaches include the information about variances and the desired probability of a correct selection and adjust the overall number of samples accordingly. A comparison of three state-of-the-art selection procedures can be found in Branke et al. (2005).

In this paper, we use a procedure that assigns a fixed total number of samples to candidate solutions, but sequentially decides how to allocate the samples on the different candidate solutions. A recently suggested sequential approach that falls into this category is the *optimal computing budget allocation* (OCBA) (Chen et al., 2000). OCBA is based on Bayesian statistics and aims at maximizing the *Approximate Probability of Correct Selection* (APCS), a lower bound for the probability of correct selection $P(\mathrm{C}S)$. It is defined as

$$\mathrm{APCS} = 1 - \sum_{i=1, i\neq b}^{k} P[\bar{X}_b > \bar{X}_i] \leq P(\mathrm{C}S),$$

where k is the number of solutions considered and b denotes the solution with the smallest sample mean performance, and $\bar{X}_i$ denotes the sample mean for solution i.

Chen et al. (2000) show that for a fixed total number of samples $T = \sum_{i=1}^{k} N_i$, the APCS can be asymptotically maximized if

$$\frac{N_i}{N_j} = \frac{\sigma_i/(\bar{X}_i - \bar{X}_b)}{\sigma_i/(\bar{X}_j - \bar{X}_b)}, i, j \in 1, 2, \ldots, k, \text{ and } i \neq j \neq b \tag{1.2}$$

and

$$N_b = \sigma_b \sqrt{\sum_{i=1, i\neq b}^{k} \frac{{N_i}^2}{{\sigma_i}^2}} \tag{1.3}$$

with σ_i being the standard deviation of the samples for solution i.

Based on these propositions, OCBA draws samples iteratively until the computational budget is exhausted.

1 Initialization: Draw n_0 initial samples for each solution. Set $l = 0, N_1^l = N_2^l = \cdots = N_k^l = n_0$, and $T = T - kn_0$

2 WHILE $T > 0$ DO

   (a) Set $l = l + 1$. Increase the computing budget by Δ_l (i.e., number of additional simulations in this iteration) and compute the new budget allocation to approximate Eq. 1.2 and Eq. 1.3.

(b) Draw additional $\max(0, N_i^l - N_i^{l-1})$ samples for each solution $i, i = 1, 2, \ldots, k$

(c) $T = T - \Delta_l$.

3 Return the index b of the system with the lowest mean $\bar{X}_b$, where $\bar{X}_b = \min_{1 \leq i \leq k} \bar{X}_i$.

For a more detailed description of OCBA, the reader is referred to Chen et al. (2000).

4. Experiments on the noisy sphere

Experimental setup

Sequential parameter optimization (SPO) is an algorithmical procedure to adjust the exogenous parameters of an algorithm, the so-called *algorithm design*, and to determine good *tuned* parameter settings for optimization algorithms (Bartz-Beielstein, 2006). It combines methods from computational statistics, *design and analysis of computer experiments* (DACE), and exploratory data analysis to improve the algorithm's performance and to understand why an algorithm performs poorly or well. SPO provides a means for reasonably fair comparisons between algorithms, allowing each algorithm the same effort and mechanism to tune parameters. Table 1.1 presents an algorithm design for PSO algorithms. The seven parameters were tuned during the SPO procedure. Details of this tuning procedure are presented in Blum (2005) and Bartz-Beielstein (2006).

In our experiments, we use the 10-dimensional sphere ($\min y = \sum_i^{10} x_i^2$) as test problem, because in this unimodal environment, the algorithm can easily find the optimum, and if it does not, this can be directly attributed to the noise. We consider additive ($\tilde{y} = y + \epsilon$) and multi-

Table 1.1. Algorithm design of the PSO algorithm. Similar designs were used in Shi and Eberhart (1999) to optimize well-known benchmark functions

Symbol	Parameter	Range	Default
s	Swarm size	$\mathbb{N}$	40
c_1	Cognitive parameter	$\mathbb{R}_+$	2
c_2	Social parameter	$\mathbb{R}_+$	2
$w_{\max}$	Starting value of the inertia weight w	$\mathbb{R}_+$	0.9
w_{scale}	Final value of w in percentage of $w_{\max}$	$\mathbb{R}_+$	0.4
$w_{\text{iterScale}}$	Percent iterations with reduced $w_{\max}$	$\mathbb{R}_+$	1.0
$v_{\max}$	Max. value of the step size (velocity)	$\mathbb{R}_+$	100

plicative ($\tilde{y} = y(1 + \epsilon)$) noise, where ϵ denotes a normally distributed random variable with mean zero and standard deviation σ. A broad range of noise levels σ was used to analyze the behavior of the algorithms and to detect the highest noise level PSO was able to cope with. For example, the experiments with additive noise used σ values from the interval $[10^{-4}, 10^2]$. All particle positions were initialized to $x_i = 10$ in all dimensions.

Unless specified otherwise, the best function value found after 10,000 function evaluations was used as a performance measure, because (i) a fixed number of evaluations is a quite fair and comparable criterion, as it does not depend on programming skills, hardware, etc., and (ii) many real-world optimization problems require simulation runs that are computationally expensive compared to the computational effort of the optimization algorithm itself.

Performance in noisy environments

For the experiments in this subsection we used the following fixed parameter settings which have proven reasonable in preliminary tests and have also been used, e.g., in Shi and Eberhart (1999): $s = 20, c_1 = c_2 = 2, w_{\max} = 0.9, w_{\text{scale}} = 4/9, v_{\max} = 100$. Figure 1.1 shows the convergence curves (fitness over time) of the standard PSO algorithm for different levels of additive noise. As can be seen, while the PSO without noise keeps improving, the algorithm stagnates in noisy environments, with the fitness level reached depending on the noise level. It is interesting to note that the performance of PSO before reaching the stagnation level is almost unaffected by the noise, indicating that a certain noise level is tolerated before the system breaks down.

While for the case of additive noise, the influence of the noise on the performance was quite regular, the effect of multiplicative noise was quite different. Figure 1.2 shows the final solution quality obtained depending on the noise level. For low levels of noise, the algorithm is basically unaffected, as the noise scales with the fitness values and most decisions can be made correctly throughout the run. This confirms the observations made in Parsopoulos and Vrahatis (2002) regarding the robustness of PSO with respect to proportional noise. On the other hand, if the noise exceeds a certain threshold, the algorithm may actually diverge and end up with solutions worse than the initial solutions. This may happen if the worse solutions have a much higher noise, making it likely that the worse fitness is accidentally compensated by an overvaluation.

When comparing the performance of PSO to a simple (1+1)-*evolution strategy* (ES), we observed a much faster progress of the ((1+1)-ES) on

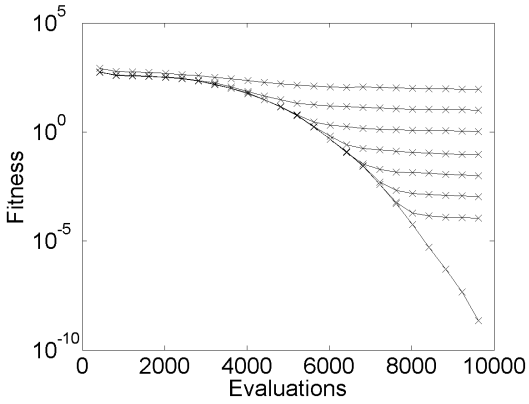

Figure 1.1. Convergence curves for the standard PSO for different levels of additive noise (from *top to bottom*, $\sigma = 100, 10, 1, 0.1, 10^{-2}, 10^{-3}, 10^{-4}, 0$).

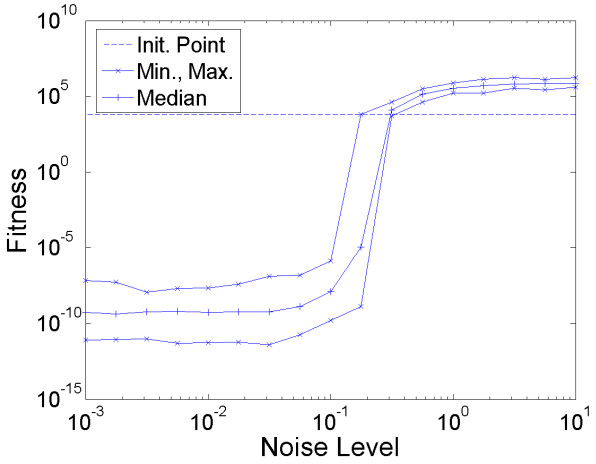

Figure 1.2. Fitness after 10,000 evaluations for various noise levels. *Horizontal line* indicates initial fitness.

the Sphere function during the first phase of the optimization. However, the algorithm encountered the same problem as the PSO: the stagnation on a certain level. Having tuned the parameters of both algorithms using SPO, we observed that the (1+1)-ES stagnated on a higher fitness level than the PSO, possibly due to an inherent advantage of population-based approaches in noisy environments.

Global versus local certainty

As described above, noise can affect PSO in two steps: when each particle chooses its local best, and when the global best is determined. In order to find out which of the two steps is more critical for the algorithm's performance, we tested two special variants of the PSO algorithm.

Variant PSO_{pc} was given the correct information whenever a particle decided between its new position and the old personal best. This variant was able to find significantly better results than the variant $\text{PSO}_{\text{default}}$. Moreover, it did not stagnate at certain fitness levels and showed a progress during the whole optimization process. Similar results were observed for multiplicative noise. The search was not misguided by the noise and for all noise levels the solutions obtained were better than the initial point.

Variant PSO_{gc} was provided with the information which of the particles' presumed best was the true best, i.e., it could correctly select the global best from all the presumed local best. In the presence of additive noise the variant could find better solutions than the $\text{PSO}_{\text{default}}$, but the optimization stagnated and could not converge to the minimum. Experiments with multiplicative noise also showed an improvement compared to $\text{PSO}_{\text{default}}$, but again the basic problem remained: the search was misled.

Overall, in our experiments with additive and multiplicative noise models, PSO_{pc} showed clearly superior performance compared to the PSO_{gc} variant. However, we have to keep in mind that variant PSO_{pc} received in each iteration the knowledge for a number of decisions, which was equal to the swarm size. In contrast, variant PSO_{gc} could decide once per iteration correctly. Furthermore, PSO_{gc} could potentially loose the true global best, namely when the decision on the first level was wrong. This could not happen with PSO_{pc}.

Parameter tuning vs. multiple evaluations

Now let us examine whether parameter tuning is sufficient for PSO to cope with the noise, or whether multiple evaluations are necessary. Results are summarized in Table 1.2. Thereby, the PSO_{rep} variant uses

a fixed number of repeated function evaluations (5 in the default setting, and this parameter is optimized by SPO as well). As can be seen, while parameter tuning improves the final solution quality for the single-evaluation and multiple-evaluation case, better results can be obtained when allowing multiple evaluations rather than only optimizing parameters and letting the algorithm cope with noise.

Integrating OCBA into PSO

Now we examine how PSO can benefit from integrating a sequential sampling procedure like OCBA (see Section 2). The variant PSO_{OCBA} uses the OCBA procedure to search for the swarm's global best among the set of positions considered in the iteration (i.e., all new positions and all local best). With the design of the PSO_{OCBA} algorithm, we aim at two objectives: (i) an increased probability to select the swarm's global best correctly, (ii) an increased probability to select the particles' personal bests correctly (as a byproduct of the repeated function evaluations of candidate positions by the OCBA method). In accordance with the OCBA technique, the position with the resulting lowest mean of the function values is selected as the swarm's global best. The new personal bests of the particles result from the comparison between the function value means of their old personal best and new positions. Function values from previous generations were stored for re-use in the next generation.

The results in Table 1.2 indicate that variant PSO_{OCBA} with an improved algorithm design generated by SPO significantly outperformed the other algorithm variants optimizing the Sphere function disturbed by additive noise. OCBA enables the algorithm to distinguish smaller function value differences than the other variants. This seems obvious with respect to $\text{PSO}_{\text{default}}$, as it has no noise reduction mechanism, but it also reaches a lower stagnation level than PSO_{rep}. OCBA's flexible assignment of samples seemed to be an advantage in the selection process. Furthermore, as OCBA allocates more samples to promising solutions, and these are more likely to survive to the next iteration, the total number of function evaluations used for decisions in one iteration (new and preserved) is higher for the OCBA variant than if each position had received the same amount of function evaluations. In fact, in our experiments we observed this number to be about twice as high, allowing more accurate decisions.

Table 1.2. Comparison of the standard algorithm *(standard)*, a variant with multiple evaluations *(rep)* and the new variant with integrated OCBA *(OCBA)*. Each variant has been tested with default *(default)* and optimized (*SPO*) parameter settings. Results ± standard error for the sphere with additive noise ($\sigma = 10$). Varying the noise level σ lead to similar results that are reported in Blum (2005). Smaller values are better.

	Parameter settings	
Algorithm	Default	SPO
$\text{PSO}_{\text{standard}}$	9.08 ± 0.43	6.94 ± 0.30
PSO_{rep}	7.59 ± 0.35	4.99 ± 0.27
PSO_{OCBA}	6.81 ± 0.67	$\mathbf{1.98 \pm 0.11}$

5. Summary and outlook

We have examined the influence of noise on the performance of PSO, and compared several algorithmic variants with default and tuned parameters. Based on our results, we make the following conclusions:

- Additive noise leads to a stagnation of the optimization process, multiplicative noise can even lead to divergence.
- Parameter tuning alone cannot eliminate the influence of noise.
- Sequential selection procedures such as OCBA can significantly improve the performance of particle swarm optimization in noisy environments. Local information plays an important role in this selection process and cannot be omitted.

Why did sequential selection methods improve the algorithm's performance? First, the selection of the swarm's best of one iteration was correct with a higher probability compared to reevaluation approaches. Second, as more samples were drawn for promising positions, positions that remained and reached the next iteration were likely to have received more samples than the average. Samples accumulated and led to a greater sample base for each iteration's decisions. These two advantages might be transferable to other population-based search heuristics, in which individuals can survive several generations. Summarizing, we can conclude that it was not sufficient to only tune the algorithm design (e.g., applying SPO), or to only integrate an advanced sequential selection procedure (e.g., OCBA). The highest performance improvement was caused by the combination of SPO and OCBA.

However, our experiments were restricted to artificial test functions only. The application of the PSO_{OCBA} variant to real-world problems

will be the next step. In such problems, the noise might not be normally distributed. First experiments, which analyzed the applicability of OCBA to an elevator group control problem proposed in Markon et al. (2006) produced promising results. Noise-dependent, variable swarm sizes as proposed in Bartz-Beielstein and Markon (2004) for evolution strategies, might improve the convergence velocity. Furthermore, it might be interesting to replace the current OCBA by sampling techniques with variable stopping rules, where the number of samples allocated per generation is not fix but depends on the confidence in the decisions.

Bibliography

Bartz-Beielstein, T. (2006). *Experimental Research in Evolutionary Computation—The New Experimentalism*. Natural Computing Series. Springer, Berlin, Heidelberg, New York.

Bartz-Beielstein, T. and Markon, S. (2004). Tuning search algorithms for real-world applications: A regression tree based approach. In Greenwood, G. W., editor, *Congress on Evolutionary Computation*, volume 1, pages 1111–1118. IEEE Press.

Bechhofer, R. E., Santner, T. J., and Goldsman, D. M. (1995). *Design and Analysis of Experiments for Statistical Selection, Screening, and Multiple Comparisons*. Wiley, New York.

Beyer, H.-G. (2001). *The Theory of Evolution Strategies*. Springer, Berlin, Heidelberg, New York.

Blum, D. (2005). Particle Swarm Optimization für stochastische Probleme. Interner Bericht der Systems Analysis Research Group SYS–2/05, Universität Dortmund, Fachbereich Informatik.

Branke, J., Chick, S., and Schmidt, C. (2005). New developments in ranking and selection: An empirical comparison of the three main approaches. In Kuhl, N., Steiger, M. N., Armstrong, F. B., and Joines, J. A., editors, *Winter Simulation Conference*, pages 708–717. IEEE.

Branke, J. and Schmidt, C. (2004). Sequential sampling in noisy environments. In *Parallel Problem Solving from Nature*, volume 3242 of *LNCS*, pages 202–211, Berlin, Heidelberg, New York. Springer.

Buchholz, P. and Thümmler, A. (2005). Enhancing evolutionary algorithms with statistical sselection procedures for simulation optimization. In Kuhl, N., Steiger, M. N., Armstrong, F. B., and Joines, J. A., editors, *Winter Simulation Conference*, pages 842–852. IEEE.

Chen, C.-H., Lin, J., Yucesan, E., and Chick, S. E. (2000). Simulation budget allocation for further enhancing the efficiency of ordinal optimization. *Discrete Event Dynamic Systems: Theory and Applications*, 10(3):251–270.

Jin, Y. and Branke, J. (2005). Evolutionary optimization in uncertain environments—a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317.

Krink, T., Filipic, B., Fogel, G. B., and Thomsen, R. (2004). Noisy optimization problems - a particular challenge for differential evolution? In *Congress on Evolutionary Computation*, pages 332–339. IEEE Press.

Liu, B., Wang, L., and Jin, Y. (2005). Hybrid particle swarm optimization for flow shop scheduling with stochastic processing time. In Hao, Y. et al., editors, *Computational Intelligence and Security*, volume 3801 of *LNAI*, pages 630–637, Berlin, Heidelberg, New York. Springer.

Markon, S., Kita, H., Kise, H., and Bartz-Beielstein, T., editors (2006). *Modern Supervisory and Optimal Control with Applications in the Control of Passenger Traffic Systems in Buildings*. Springer, Berlin, Heidelberg, New York.

Parsopoulos, K. E. and Vrahatis, M. N. (2001). Particle swarm optimizer in noisy and continuously changing environments. In Hamza, M., editor, *Artificial Intelligence and Soft Computing*, pages 289–294. IASTED/ACTA Press.

Parsopoulos, K. E. and Vrahatis, M. N. (2002). Particle swarm optimization for imprecise problems. In Fotiadis, D. and Massalas, C., editors, *Scattering and Biomedical Engineering, Modeling and Applications*, pages 254–264. World Scientific.

Rudolph, G. (1997). Reflections on bandit problems and selection methods in uncertain environments. In Bäck, T., editor, *International Conference on Genetic Algorithms*, pages 166–173. Morgan Kaufmann.

Schmidt, C., Branke, J., and Chick, S. (2006). Integrating techniques from statistical ranking into evolutionary algorithms. In Rothlauf, F. et al., editors, *Applications of Evolutionary Computation*, number 3907 in LNCS, pages 752–763, Berlin, Heidelberg, New York. Springer.

Shi, Y. and Eberhart, R. (1998). Parameter selection in particle swarm optimization. In Porto, V., Saravanan, N., Waagen, D., and Eiben,

A., editors, *Evolutionary Programming*, volume VII, pages 591–600. Springer, Berlin, Heidelberg, New York.

Shi, Y. and Eberhart, R. (1999). Empirical study of particle swarm optimization. In Angeline, P. J. et al., editors, *Congress on Evolutionary Computation*, volume 3, pages 1945–1950.