Particle Swarm Optimization and Sequential Sampling in Noisy Environments

Thomas Bartz-Beielstein* Daniel Blum* Jürgen Branke†

*Department of Computer Science, University of Dortmund
Joseph-von-Fraunhofer-Straße 20, D-44221 Dortmund, Germany
{thomas.bartz-beielstein,daniel.blum}@udo.edu

†Institute AIFB, University of Karlsruhe
Englerstraße 11, D-76128 Karlsruhe, Germany
branke@aifb.uni-karlsruhe.de

1 Introduction

In many real-world optimization problems, function values can only be estimated but not determined exactly. 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 the analyses presented in [1] and [2] by also examining the influence of algorithm parameters, by considering a wider spectrum of noise levels, and analysing 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 [3, 4, 5], but their application to the PSO algorithm is new.

The paper is structured as follows. First, we briefly introduce PSO in Sec. 2. Then, the effects of noise and sequential sampling techniques are discussed in Sec. 3. Sec. 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.

2 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 i-th 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. The term p_g^* denotes the best position of the whole swarm. Velocities and positions are updated for each dimension $1 \leq d \leq n$ as follows:

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

$$x_i(t+1) = x_i(t) + v_i(t+1). \quad (2)$$

Before optimization can be started, several parameters or factors have to be specified. These so-called *exogenous* factors will be analyzed in the remainder of this article. 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, which regulate the influence of the local and the global information. The factor w in the *momentum* term of Equation 1 is called *inertia weight*. It was introduced in [6] to balance the global and local search abilities.

3 How to cope with noise

Noise makes it difficult to compare different solutions and select the better ones. In PSO, noise affects two operations: In every iteration, (i) each particle has to compare the new solution to its previous best and retain the better one, and (ii) the overall best solution found so far has to be determined. Wrong decisions can cause a *stagnation* of the search process: Over-valuated 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. How strongly the noise affects the overall performance of PSO is not clear. For example for evolution strategies, it has been shown that increasing the population size may help the algorithm to cope with the noise [7]. A comprehensive overview of various evolutionary algorithm variants in the presence of noise is given in [8]. Thus, one might hope that also PSO can somehow cope with the noise, and that it is sufficient to adjust its parameters.

On the other hand, 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. This reduces the standard deviation of the noise by a factor of $\sqrt{n}$, while increasing the running time by a factor of n.

More sophisticated sampling approaches include the information about variances and the desired probability of a correct selection and adapt the number of samples according to this information. Two-stage procedures use the first stage is used to estimate 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. We consider procedures that assign a fixed total number of samples to candidate solutions, but sequentially decide how to allocate the samples on the different candidate solutions. 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. A recently suggested sequential approach is the *optimal computing budget allocation* (OCBA) [9]. The aim of this method is to find the best within a set of candidate solutions and to select it with a high probability. The OCBA variant we are using here is a closed procedure and can be used for unknown and unequal variances. It draws samples sequentially until the computational budget is exhausted while adjusting the selection of samples to maximize the probability of a correct selection.

4 Experiments

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 [10]. 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.

In our experiments, we use the 10-dimensional sphere ($y = \sum_i^{10} x_i^2$) 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 multiplicative ($\tilde{y} = y(1 + \epsilon)$) 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 $[10^{-4}, 10^2]$.

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.

4.1 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 algorithm performance, we made a preliminary study with 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 $PSO_{default}$. Moreover, it did not stagnate at certain fitness levels and showed a progress during the whole optimization process. Similar results were observed for the 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 $PSO_{default}$, but the optimization stagnated and could not converge to the minimum. Experiments with multiplicative noise also showed an improvement compared to $PSO_{default}$, 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}.

4.2 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 Tab. 1. 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 the noise.

4.3 Integrating OCBA into PSO

We examine how PSO can benefit from integrating a sequential sampling procedure like OCBA (see Sec. 3). 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. All function evaluations made for

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

Table 1: *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* $\pm$ *standard error for the sphere with additive noise* ($\sigma = 10$)*.*

the new personal best position are stored for re-use in the next generation.

The results in Tab. 1 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 $PSO_{default}$, as it has no noise reduction mechanism, but also performs better than PSO_{rep}. The PSO_{OCBA} could evaluate more candidate solutions than the other two variants, because it needed less evaluations per position and was able to reach the lowest stagnation level. 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.

Additionally to the experimental results presented so far, other heuristics have been analyzed, too. We have observed a much faster progress of a two-membered evolution strategy ((1+1)-ES) on the Sphere function during the first phase of the optimization. Nevertheless, 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.

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: (i) Parameter tuning alone cannot eliminate the influence of noise. (ii) 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 [10] produced promising results. Furthermore, it might be interesting to include other sequential sampling techniques, e.g. sequential selection with memory as proposed in [11], in our investigations.

References

[1] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimizer in noisy and continuously changing environments," in *Artificial Intelligence and Soft Computing*, M. Hamza, Ed. IASTED/ACTA Press, 2001, pp. 289–294.

[2] T. Krink, B. Filipic, G. B. Fogel, and R. Thomsen, "Noisy optimization problems - a particular challenge for differential evolution?" in *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*. Portland, Oregon: IEEE Press, 20-23 June 2004, pp. 332–339.

[3] G. Rudolph, "Reflections on bandit problems and selection methods in uncertain environments," in *Proc. Seventh Int'l Conf. Genetic Algorithms (ICGA'97), East Lansing MI*, T. Bäck, Ed. San Francisco CA: Morgan Kaufmann, 1997, pp. 166–173.

[4] T. Bartz-Beielstein and S. Markon, "Tuning search algorithms for real-world applications: A regression tree based approach," in *Proc. 2004 Congress on Evolutionary Computation (CEC'04), Portland OR*, G. W. Greenwood, Ed., vol. 1. Piscataway NJ: IEEE Press, 2004, pp. 1111–1118.

[5] J. Branke and C. Schmidt, "Sequential sampling in noisy environments," in *Parallel Problem Solving from Nature*, ser. LNCS. Springer, 2004.

[6] Y. Shi and R. Eberhart, "Parameter selection in particle swarm optimization," in *Evolutionary Programming*, V. Porto, N. Saravanan, D. Waagen, and A. Eiben, Eds. Springer, 1998, vol. VII, pp. 591–600.

[7] H.-G. Beyer, *The Theory of Evolution Strategies*. Springer, Heidelberg, 2001.

[8] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments—a survey," *IEEE Transactions on Evolutionary Computation*, 2005.

[9] J. Chen, C. Chen, and D. Kelton, "Optimal computing budget allocation of indifference-zone-selection procedures," 2005, working paper, taken from http://www.cba.uc.edu/faculty/keltonwd/, Jan. 2005.

[10] T. Bartz-Beielstein, "New experimentalism applied to evolutionary computation," Ph.D. dissertation, University of Dortmund, April 2005.

[11] S.-H. Kim and B. L. Nelson, "A fully sequential procedure for indifference-zone selection in simulation," *ACM Trans. Model. Comput. Simul.*, vol. 11, no. 3, pp. 251–273, 2001.