Analysis of Particle Swarm Optimization Using Computational Statistics

T. Bartz–Beielstein[*1], **K.E. Parsopoulos**[**2], and **M.N. Vrahatis**[***2]

[1] Department of Computer Science XI, University of Dortmund, D–44221 Dortmund, Germany

[2] Department of Mathematics, University of Patras, GR–26110 Patras, Greece

Received xxxx, revised xxxx, accepted xxxx

We propose a new methodology for the experimental analysis of evolutionary optimization algorithms. The proposed technique employs computational statistic methods to investigate the interactions among optimization problems, algorithms, and environments. The technique is applied for the parameterization of the Particle Swarm Optimization algorithm. An elevator supervisory group control system is introduced as a test case to provide intuition regarding the performance of the proposed approach in highly complex real–world problems.

1 Introduction

Modern search heuristics have proved to be very useful for solving complex real–world optimization problems that cannot be tackled through classical optimization techniques [1]. Many of these search heuristics involve a set of exogenous parameters that affect their convergence properties. An optimal parameter setting depends on the problem at hand as well as on the restrictions posed by the environment (i.e., time and hardware constraints).

Particle Swarm Optimization (PSO) is a swarm intelligence optimization algorithm [2]. The main inspiration behind PSO was the flocking behavior of swarms and fish schools. PSO has proved to be very efficient in numerous applications in science and engineering [3, 4, 5, 6]. PSO's convergence is controlled by a set of parameters that are usually either determined empirically or set equal to widely used default values.

We suggest an approach for determining the PSO parameters, tailored for the optimization problem at hand (we consider only minimization cases, although the technique can be straightforwardly applied in maximization problems). The proposed approach employs techniques from computational statistics and statistical experimental design, and it is applicable on all PSO variants. It can be also applied to any parameterizable search algorithm, such as evolutionary algorithms (EA) or other direct search methods. To justify the usefulness of our approach, we analyze the properties of PSO from the viewpoint of an optimization practitioner in the context of a real–world optimization problems. More specifically, we consider the optimization of an elevator group controller as well as well–known test functions, extending the approaches proposed in [7] and [8].

[*] Corresponding author: e-mail: thomas.bartz-beielstein@udo.edu, Phone: +00 49 231 9700977, Fax: +00 49 231 9700959
[**] e-mail: kostasp@math.upatras.gr, Phone: +30 2610 997348, Fax: +30 2610 992965
[***] e-mail: vrahatis@math.upatras.gr, Phone: +30 2610 997374, Fax: +30 2610 992965

2 Problem, Algorithm and Environment

Almost all new office buildings in modern cities are high rise buildings and require efficient elevator systems. The elevator group controller assigns elevators to customer service calls, based on a specific policy. This policy should be optimal with respect to different goals, such as the overall throughput, waiting times, energy consumption etc. Different building types and traffic situations make controllers and the related policies incomparable. This was the main motivation for the development of a simplified elevator group control model, which is called the *Sequential Ring* (S–ring). The state of the system at time t is mapped to a binary vector. Since the S–ring has only a few parameters, it can be used as a test problem generator for benchmark testing of algorithms[1] [9, 10]. The S–ring constitutes also a well suited model to simulate and analyze bunching problems [11].

PSO belongs to the class of stochastic, population–based optimization algorithms [12]. It exploits a population of individuals to probe the search space. In this context, the population is called a *swarm* and the individuals are called *particles*. Each particle moves with an adaptable velocity within the search space, and it retains the best position it has ever visited. PSO is a stochastic algorithm, and, therefore, it requires random number seeds. An optimization practitioner is interested in robust solutions, i.e., solutions independent from the random seeds that are used to generate the random numbers. The proposed statistical methodology provides guidelines to design robust PSO algorithms under restrictions, such as a limited number of function evaluations and processing units. These restrictions can be modeled by considering the performance of the algorithm in terms of the (expected) best function value for a limited number of fitness function evaluations. A discussion on different problem classes for real–world optimization problems is provided in [13].

3 Computational Statistics

Statistical methods, such as experimental design techniques and regression analysis, can be used to analyze the experimental setting introduced in the previous section. Breiman *et al.* introduced regression trees as a "flexible non–parametric tool to the data analyst's arsenal" [14]. Regression trees are used for screening variables and checking the adequacy of regression models [15]. The construction of classification and regression trees (CART) can be seen as a type of variable selection, which is similar to the stepwise regression techniques in classical regression analysis [16]. Compared to linear models, tree–based models are easier to interpret when qualitative and quantitative predictors appear in the model.

A fractional factorial design was considered to perform the regression tree based screening experiments with PSO [17]. We considered a sequential approach that combines existing as well as new results, and enables a step–wise increase in the regression model complexity. Starting with a simple linear model, the final model can be analyzed with response surface methods (RSM). Furthermore, design and analysis of computer experiments (DACE) methods for the analysis of optimization algorithms, as proposed in [18], have been used. Santner *et al.* [19] presented a heuristic algorithm for unconstrained global optimization problems, which is based on the expected improvement [20]. The discussion in [19] leads to the conclusion that new designs are attractive if either there is a hight probability that their predicted output is below the current observed minimum and/or there is a large uncertainty in the predicted output. This result comes in line with the experimenters' intention to avoid sites that guarantee worse results, and constituted the motivation for the following heuristic [18]:

1. Choose an initial design $\mathcal{D}_n$ with n points.
2. Run the algorithm at $\vec{x}_i \in \mathcal{D}_n$, $i = 1, \ldots, n$, to obtain the vector of output values $\vec{y}(\vec{x})$.
3. Check the termination criterion.

[1] A reference implementation of S–ring model can be requested from the authors: `thomas.bartz-beielstein@udo.edu`.

Table 1 Tuning of the PSO parameters on the S–ring with DACE. First, second and third design with corresponding fitness values Y.

Y	P	c_1	c_2	$w_{\max}$	w_{Scale}	$w_{\text{IterScale}}$	$V_{\max}$
2.4626	8	1.74419	2.11749	0.842983	0.444293	0.951067	36.7564
2.4237	6	2.24586	2.44838	0.897144	0.606148	0.812619	118.051
2.4167	6	2.2722	1.71572	0.937604	0.561817	0.64614	138.371

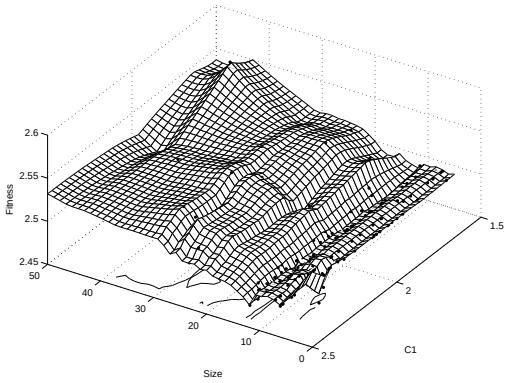

Fig. 1 DACE fit. PSO optimizing the S–ring model.

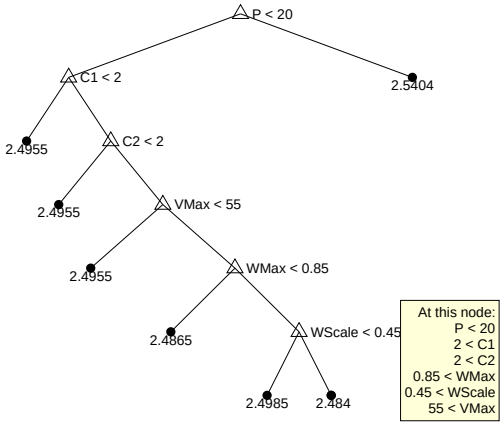

Fig. 2 Regression tree showing the parameterization of the PSO optimizing the S–ring model.

4. Select a new point $\vec{x}_{n+1}$ that maximizes the expected improvement (cf. [20]).
5. Run the algorithm at $\vec{x}_{n+1}$ to obtain the output $y(\vec{x}_{n+1})$.
6. Set $\mathcal{D}_{n+1} = \mathcal{D}_n \cup \{\vec{x}_{n+1}\}$, $n = n + 1$, and go to 3.

4 Experimental Results

We first investigated simple well–known test functions [21] to gain an intuition regarding the workings of the proposed technique. For example, for the 10–dimensional Rosenbrock function and the parameterizations provided in [21], we obtained an improved PSO parameterization that consists of a population size $P = 44$, $c_1 = 1.60379$, $c_2 = 2.19376$, $w_{\max} = 0.825048$, $w_{\text{Scale}} = 0.312031$, $w_{\text{IterScale}} = 0.932467$, and $V_{\max} = 102.235$. The mean best function value was 9.8118 (with a minimum value equal to 0.0037649), while, in [21], the mean best function value was equal to 96.1715.

In the next step of our analysis, the S–ring model was considered. The improved parameterization using DACE was $P = 6$, $c_1 = 2.2722$, $c_2 = 1.71572$, $w_{\max} = 0.937604$, $w_{\text{Scale}} = 0.561817$, $w_{\text{IterScale}} = 0.64614$, and $V_{\max} = 138.371$, and the best function value was 2.4167. Furthermore, small swarms proved to perform better than large ones. The DACE and CART techniques provided similar configurations of the PSO parameters, as depicted in Figs. 1 and 2.

5 Synopsis

CART and DACE provide effective and efficient means to improve PSO performance, significantly. Only a few tree growing phases and DACE iterations (less than 5) were needed to find better PSO parameterizations. Regression trees have shown their ability to model the dependencies between different PSO

parameterizations in a very intuitive manner. Although they are more conservative than the classical regression techniques, their results are reliable and similar to the results obtained from classical regression analysis and DACE. Moreover, classical regression models, which are much more complicated and require assumptions on the underlying distribution, can be alleviated. On the other hand, DACE are restricted to quantitative factors, but provide a more detailed insight into the influence of different factors on the algorithm's performance. A drawback of the proposed approach, which is common to all statistical methods in this field, is the determination of a good starting design.

Acknowledgements T. Bartz–Beielstein's research was supported by the DFG as a part of the collaborative research center "Computational Intelligence" (531). We acknowledge the partial support of the "Pythagoras" research grant awarded by the Greek Ministry of Education and Religious Affairs and the European Union.

References

[1] H.-P. Schwefel, I. Wegener, and K. Weinert, editors. *Advances in Computational Intelligence – Theory and Practice*. Natural Computing Series. Springer, Berlin, 2003.
[2] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proceedings IEEE International Conference on Neural Networks*, volume IV, pages 1942–1948, Piscataway, NJ, 1995. IEEE Service Center.
[3] K. E. Parsopoulos and M. N. Vrahatis. On the computation of all global minimizers through particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):211–224, 2004.
[4] K. E. Parsopoulos and M. N. Vrahatis. Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, 1(2–3):235–306, 2002.
[5] K. E. Parsopoulos, E. I. Papageorgiou, P. P. Groumpos, and M. N. Vrahatis. Evolutionary computation techniques for optimizing fuzzy cognitive maps in radiation therapy systems. *Lecture Notes in Computer Science (LNCS)*, 3102:402–413, 2004.
[6] N. G. Pavlidis, K. E. Parsopoulos, and M. N. Vrahatis. Computing Nash equilibria through computational intelligence methods. *Journal of Computational and Applied Mathematics*, 2004. in press.
[7] T. Beielstein, K. E. Parsopoulos, and M. N. Vrahatis. Tuning PSO parameters through sensitivity analysis. Tech. Report CI 124/02, Department of Computer Science, University of Dortmund, Dortmund, Germany, 2002.
[8] T. Bartz-Beielstein, M. de Vegt, K.E. Parsopoulos, and M.N. Vrahatis. Designing particle swarm algorithms with regression trees. Technical Report of the Collaborative Research Center 531 *Computational Intelligence* CI–173/04, University of Dortmund, December 2001.
[9] S. Markon, D.V. Arnold, T. Bäck, T. Beielstein, and H.-G. Beyer. Thresholding – a selection operator for noisy ES. In J.-H. Kim, B.-T. Zhang, G. Fogel, and I. Kuscu, editors, *Proc. 2001 Congress on Evolutionary Computation (CEC'01)*, pages 465–472, Seoul, Korea, May 27–30, 2001. IEEE Press, Piscataway NJ.
[10] T. Beielstein, S. Markon, and M. Preuß. Algorithm based validation of a simplified elevator group controller model. In T. Ibaraki, editor, *Proc. 5th Metaheuristics Int'l Conf. (MIC'03)*, pages 06/1–06/13 (CD–ROM), Kyoto, Japan, 2003.
[11] G. Barney. *Elevator Traffic Analysis, Design and Control*. Cambridge U.P., 1986.
[12] J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, 2001.
[13] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.
[14] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.
[15] T. M. Therneau and E. J. Atkinson. An introduction to recursive partitioning using the rpart routines. Technical Report 61, Department of Health Science Research, Mayo Clinic, Rochester, 1997.
[16] J. M. Chambers and T. H. Hastie, editors. *Statistical Models in S*. Wadsworth & Brooks/Cole, Pacific Grove, California, 1992.
[17] G. E. P. Box and J. S. Hunter. The 2^{k-p} fractional factorial designs, part I. *Technometrics*, 3:311–352, 1961.
[18] T. Bartz-Beielstein. Tuning search algorithms for real-world applications: A regression tree based approach. In *Proceedings of the 2004 Congress on Evolutionary Computation CEC2004*, 2004. accepted for publication.
[19] T.J. Santner, B.J. Williams, and W.I. Notz. *The Design and Analysis of Computer Experiments*. Springer, 2003.
[20] M. Schonlau, W.J. Welch, and R.D. Jones. Global versus local search in constrained optimization of computer models. In N. Flournoy, W.F. Rosenberger, and W.K. Wong, editors, *New developments and applications in experimental design*, volume 34, pages 11–25. Institute of Mathematical Statistics, 1998.
[21] Y. Shi and R.C. Eberhart. Empirical study of particle swarm optimization. In P.J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, editors, *Proceedings of the Congress of Evolutionary Computation*, volume 3, pages 1945–1950, 1999.