Analysis of Fitness Noise in Particle Swarm Optimization: From Robotic Learning to Benchmark Functions

Ezequiel Di Mario Iñaki Navarro Alcherio Martinoli

Abstract— **Population-based learning techniques have been proven to be effective in dealing with noise and are thus promising tools for the optimization of robotic controllers, which have inherently noisy performance evaluations. This article discusses how the results and guidelines derived from tests on benchmark functions can be extended to the fitness distributions encountered in robotic learning. We show that the large-amplitude noise found in robotic evaluations is disruptive to the initial phases of the learning process of PSO. Under these conditions, neither increasing the population size nor increasing the number of iterations are efficient strategies to improve the performance of the learning. We also show that PSO is more sensitive to good spurious evaluations of bad solutions than bad evaluations of good solutions, i.e., there is a non-symmetric effect of noise on the performance of the learning.**

I. Introduction

There are several sources of randomness that make performance evaluations of robotic controllers inherently noisy. In addition to the obvious sensor and actuator noise, there are other factors such as varying initial conditions, manufacturing tolerances, or changes in the environment that can increase the uncertainty in performance measurements.

Population-based learning techniques have been proven to be effective in dealing with noise in fitness evaluations [1]. Within this family of algorithms, we can find examples on the successful performance under noise for Particle Swarm Optimization [2], [3], Genetic Algorithms [4], and Evolutionary Strategies [5], [6]. Therefore, these techniques are promising tools for the design of high-performing robotic controllers.

However, with the exception of [6], most of these studies were conducted on benchmark functions with an additive Gaussian noise model only. Since adequate benchmarks help in the choice of the algorithmic variations and parametrizations to obtain the highest possible performance with the least number of function evaluations, in this article we would like to test whether the results and guidelines derived from tests on the benchmark functions can be extended to the noisy performance evaluations encountered in multi-robot learning. In order to achieve this goal, we are going to analyze the effects of noise found in a robotic learning case study, and then we will attempt to model and reproduce these effects on numerical benchmark functions with different noise distributions.

We focus this research on the PSO algorithm [7], which allows a distributed implementation in each robot, speeding up the optimization process and adding robustness to failure of individual robots. PSO has been applied to several problems in the robotics domain, such as robotic search [8], odor source localization [9], [10], and path planning [11], [12]. In particular, Pugh et al. [13] showed that PSO could outperform Genetic Algorithms on benchmark functions and for a robotic obstacle-avoidance task.

In our previous work [14], we compared PSO with Q-Learning for the same multi-robot obstacle avoidance benchmark task used in this paper, showing that both algorithms could achieve similar performances if a continuous state representation was used for Q-Learning. More recently, we have proposed guidelines to adjust the PSO algorithmic parameters in multi-robot learning, aiming to reduce the total evaluation time so that it is feasible to implement the adaptation process within the limits of the robots' energy autonomy [15].

The remainder of this article is organized as follows. Section II describes the algorithms and parameters used in this article. Section III presents a robotic learning case study where we analyze the fitness distributions and their impact on the learning process. In Section IV we model the effects of noise found on the robotic case study in two numerical benchmark functions with added noise, and discuss the differences with previous results on benchmark functions. Finally, Section V concludes the article with a summary of our findings and an outlook for future work.

II. Optimization Algorithms

In this article, two different optimization algorithms are used both for robotic learning and benchmark functions. The first one is a standard PSO version [7], while the second is a distributed, noise-resistant, averaging variation of PSO introduced by Pugh et al. [13]. This noise-resistant version (PSOavg) operates by re-evaluating personal best positions and averaging them with the previous evaluations. According to previous works and the results on this paper, it represents an improvement when dealing with noisy fitness functions. The pseudocode for PSOavg is shown in Figure 1. The difference with the standard PSO pseudocode is the addition of lines 6 and 7.

Distributed Intelligent Systems and Algorithms Laboratory, School of Architecture, Civil and Environmental Engineering, École Polytechnique Fédérale de Lausanne {ezequiel.dimario, inaki.navarro, alcherio.martinoli}@epfl.ch

This research was supported by the Swiss National Science Foundation through the National Center of Competence in Research Robotics.

```
Intialize particles
for N_i iterations do
    for N_p particles do
        Update particle position
        Evaluate particle
        Re-evaluate personal best
        Aggregate with previous best
        Share personal best
    end for
end for
```

Fig. 1. PSOavg algorithm.

TABLE I
PSO PARAMETER VALUES

Parameter	Value
Number of robots N_{rob}	4
Population size N_p	24
Iterations N_i	200
Evaluation span t_e	30 s
Re-evaluations N_{re}	1
Personal weight pw	2.0
Neighborhood weight nw	2.0
Dimension D	24
Inertia w	0.8
V_{max}	20

In PSO, the movement of particle i in dimension j depends on three components: the velocity at the previous step weighted by an inertia coefficient w, a randomized attraction to its personal best $x^*_{i,j}$ weighted by w_p, and a randomized attraction to the neighborhood's best $x^*_{i',j}$ weighted by w_n (Eq. 1). $rand()$ is a random number drawn from a uniform distribution between 0 and 1.

$$v_{i,j} = w \cdot v_{i,j} + w_p \cdot rand() \cdot (x^*_{i,j} - x_{i,j}) + w_n \cdot rand() \cdot (x^*_{i',j} - x_{i,j}) \quad (1)$$

In the robotic learning, the algorithms are implemented in a distributed fashion, which reduces the total evaluation time required by a factor equal to the number of robots. Each robot evaluates in parallel a possible candidate solution and shares the solution with its neighbors in order to create the next pool of candidate solutions. The neighborhood presents a ring topology with one neighbor on each side. Particles' positions and velocities are initialized randomly with a uniform distribution in a $[-X,X]$ interval, and their maximum velocity is also limited to that interval. In the robotic learning case this interval is $[-20,20]$, while in the benchmark functions the interval is $[-5.12,5.12]$.

The PSO algorithmic parameters for the robotic task are set following the guidelines for limited-time adaptation we presented in our previous work [15] and are shown in Table I. The same set of parameters is used for the optimization of the benchmark functions with noise.

III. Robotic Learning

One of the aims of this article is to understand and analyze the randomness in robotic performance evaluations and how it affects the learning process. We have chosen obstacle avoidance as a task to illustrate robotic learning because it is a fundamental task popular in the robotic learning literature [16]–[20], and it requires basic sensors and actuators available in most mobile robots.

We use the metric of performance introduced in [16], which was also used in [17], [19], [20]. It consists of three factors, all normalized to the interval [0, 1], which reward robots that move quickly, turn as little as possible, and stay away from obstacles.

We conduct experiments in two different environments. The first one is an empty square arena of 2 m x 2 m, where the walls and the other robots are the only obstacles. The second environment is the same bounded arena with 15 cylindrical obstacles added (diameter 10 cm). The obstacles are randomly repositioned before each fitness evaluation, meaning that the second environment is not only more complex but also variable from evaluation to evaluation, and so more noisy. The initial robots positions are set randomly with a uniform probability distribution, verifying that they do not overlap with obstacles or other robots.

All experiments are conducted with 4 Khepera III robots in simulation. The Khepera III mobile robot is a differential wheeled vehicle with a diameter of 12 cm. It is equipped with nine infra-red sensors for short range obstacle detection, which in our case are the only external inputs for the controllers. Simulations are performed in Webots [21], a realistic physics-based submicroscopic simulator that models dynamical effects such as friction and inertia.

The controller used is a recurrent artificial neural network of two units with sigmoidal activation functions. The outputs of the units determine the wheel speeds. Each neuron has 12 input connections: the 9 infrared sensors, a connection to a constant bias speed, a recurrent connection from its own output, and a lateral connection from the other neuron's output, resulting in 24 weight parameters in total. These 24 parameters define the dimensionality of the learning space of the algorithms.

More details on the experimental setup and controller can be found in our previous work [20].

A. Fitness Functions

A major challenge in comparing robotic learning algorithms that is not present in benchmark functions is that it is not possible to separate the deterministic and random components of the fitness evaluations, i.e., there is no single true fitness value for a given position. This implies that a single evaluation does not provide sufficient information on the goodness of a particular solution. Therefore, in order to test the outcome of a given optimization technique, we characterize each candidate solution by repeatedly evaluating the fitness a large number of times and look at the probabilistic distribution of those evaluations. In particular, for the results presented in this section, we do 1000 a posteriori evaluations of the candidate solutions given by the optimization algorithm.

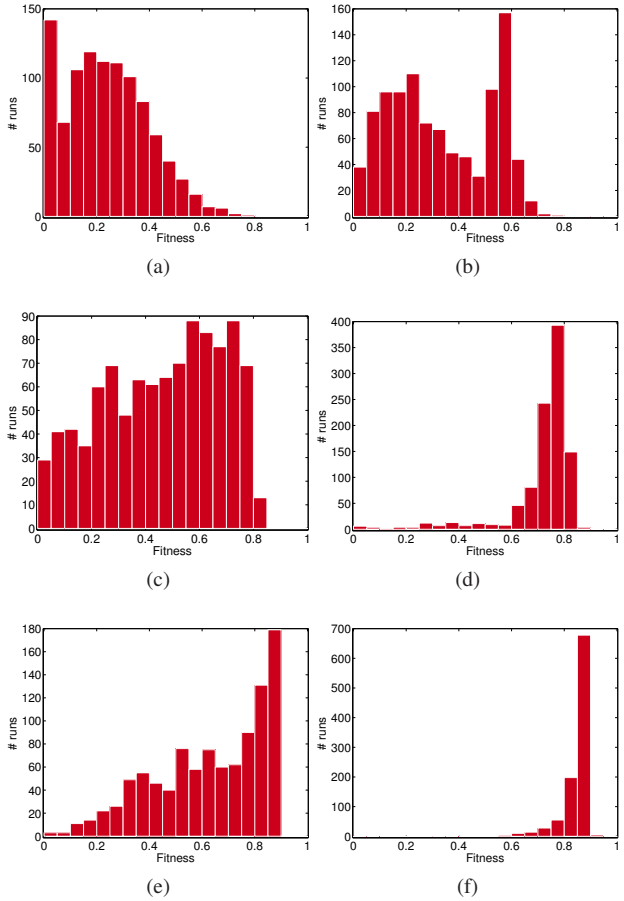


Fig. 2. Distributions of 1000 a posteriori evaluations of the fitness of the global best solution at different iterations corresponding to two PSOavg runs, one in each environment. p value of one-sample Kolmogorov-Smirnov test in parenthesis. (a) Iteration 1 in obstacles arena ($p = 0.000891$). (b) Iteration 3 in obstacles arena ($p < 10^{-6}$). (c) Iteration 11 in obstacles arena ($p = 0.000017$). (d) Iteration 20 in obstacles arena ($p < 10^{-6}$). (e) Iteration 16 in empty arena ($p < 10^{-6}$). (f) Iteration 19 in empty arena ($p < 10^{-6}$).

Figure 2 shows examples of the distributions obtained for global best positions at several iterations of PSOavg in the two aforementioned environments (with and without obstacles). The distributions shown in Figures 2a, 2b, 2c, and 2e are not gaussian according to a one-sample Kolmogorov-Smirnov statistical test. This test was performed as well for every distribution of the PSOavg learning in the two environments with negative results.

Looking at every distribution of the fitness of the solutions obtained by PSOavg learning in the two environments we see that those corresponding to the empty environment have lower standard deviations (ranges from 0.039 to 0.226, averaging 0.099 over all tested distributions) than those in the environment with obstacles (ranges from 0.115 to 0.221, averaging 0.175 over all tested distributions). This is something that could be expected from the experimental setup given the randomized placement of obstacles and robots in each evaluation.

It is important to mention that the distributions shown here and found in our analysis can not be generalized directly

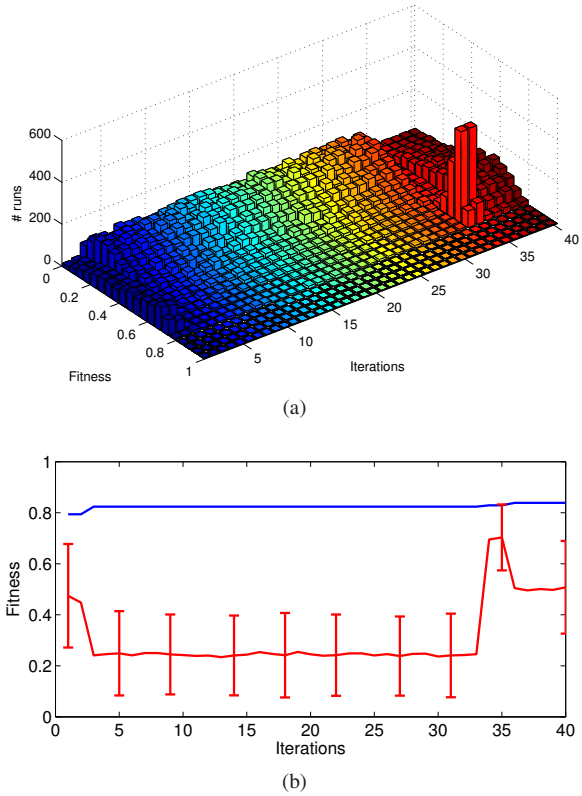


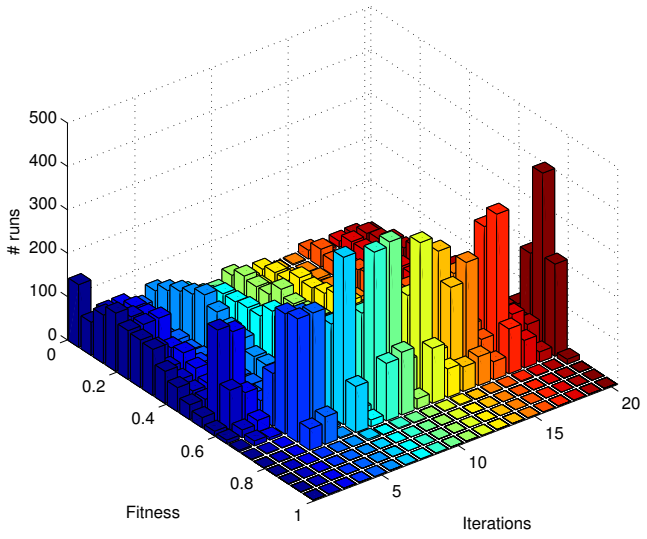
Fig. 3. Evolution of the global best fitness for a single run of standard PSO in the environment with obstacles. (a) The distributions of 1000 a posteriori evaluations of global best fitness. (b) In blue, the global best value calculated by standard PSO algorithm. In red, the average of global best fitness over 1000 a posteriori evaluations (vertical bars indicate the standard deviation).

to the whole search space, since they only correspond to positions found during the PSO learning.

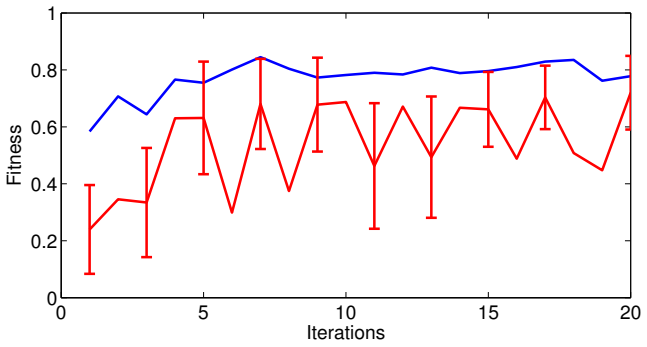
B. Learning with Noise

Figure 3a shows the distributions of the global best solutions (reevaluated 1000 times) for each iteration during a standard PSO run. In Figure 3b, in red, we can see the average value of these distributions (and the standard deviation with vertical bars). Although it only represents a single run of PSO we can see how standard PSO is actually not able to learn in a proper way, since the averaged fitness value decreases sometimes or stays stable for a long period. The global best value that PSO calculated during the optimization is shown in blue. We can see it is monotonically increasing. The problem for the learning algorithm is that these fitness values obtained with a single sample and used by PSO for those positions do not correspond with the more accurate estimations obtained by averaging the 1000 a posteriori evaluations (in red).

In order to better understand this effect, we can look back to Figure 3a to see that the distribution at iteration 3 has a large standard deviation. What happened is that the given position was evaluated by PSO with a high value (0.824), becoming the global best and staying as such until iteration 34. Since standard PSO does not re-evaluate the personal best positions this global best candidate was never filtered out.



(a)



(b)

Fig. 4. Evolution of the global best fitness for a single run of PSOavg in the environment with obstacles. (a) The distributions of 1000 a posteriori evaluations of global best fitness. (b) In blue, the global best value calculated by PSOavg algorithm. In red, the average of global best fitness over 1000 a posteriori evaluations (vertical bars indicate the standard deviation).

A similar analysis can be done for PSOavg by looking at Figure 4. For the sake of fairness in terms of total number of evaluations, we only performed 20 iterations of PSOavg, corresponding to half of the number of iterations done in standard PSO. It can be observed how, due to the reevaluation of the personal bests, there is less difference between the global best calculated by PSOavg (in blue) and its a posteriori estimation (in red) than in standard PSO. Therefore, the estimated fitness of the global bests (Figure 4b in red), increases during the learning process, which implies that the overall quality of the solutions is improving. Looking at the initial distributions of the global best solutions learned with PSOavg (Figure 4a), we notice that they have a noise profile similar to the one from standard PSO, since they correspond to the same environment with obstacles and the initial learning conditions are the same for both algorithms .

IV. Benchmark Functions

We perform PSO runs on two standard benchmarks, the sphere function (Eq. 2) and Rosenbrock's function (Eq. 3).

$$f_1(\mathbf{x}) = \sum_{i=1}^{D} x_i^2 \tag{2}$$

$$f_2(\mathbf{x}) = \sum_{i=1}^{D-1} [(1-x_i^2)+100(x_{i+1}-x_i^2)^2] \tag{3}$$

In order to reproduce the effects encountered in robotic learning, we normalize the function values to the interval $[0,1]$ by dividing by the maximum value of each function in the initial position range $x_{init} = [-5.12,\ 5.12]$, which we denote by $\max f_i$. We then add noise from two distributions: a Gaussian distribution with zero mean and standard deviation σ (Eq. 4), and a Bernoulli distribution with probability p and amplitude A (Eq. 5).

$$f_i^g(\mathbf{x}) = \frac{f_i(\mathbf{x})}{\max f_i} + \mathcal{N}(0,\sigma) \tag{4}$$

$$f_i^b(\mathbf{x}) = \frac{f_i(\mathbf{x})}{\max f_i} + A \cdot \mathcal{B}(p) \tag{5}$$

If after adding noise the function values are greater than one or less than zero, they are set to one and zero respectively to maintain the fitness bounded in the interval $[0,1]$.

The parameters for PSO on the benchmark functions are the same as the ones used for robotic learning (Table I), with the exception of two parameters that are specific to the robotic case-study and are therefore omitted in the benchmark functions: number of robots and evaluation span (one benchmark function evaluation is assumed to be equivalent to the evaluation of a controller for the whole evaluation span).

A significant difference between benchmark functions and robotic learning is that it is possible to remove the noise from the benchmark functions to see the real performance of the algorithm. Therefore, all results reported in this section show the fitness function values obtained when evaluating the functions without noise (there is no need of 1000 a posteriori evaluations since the fitness is obtained in a deterministic way). Another minor difference is that benchmark functions are minimized as opposed to maximized.

When comparing PSO and PSOavg, it should be noted that each iteration of PSOavg requires twice as many function evaluations as standard PSO due to the re-evaluations of the personal bests. Therefore, in order to maintain the total number of evaluations equal and compare both algorithms fairly, we run PSO for twice as many iterations as PSOavg. In addition, we define a *step* of an algorithm to be equal to one iteration of PSO, and half an iteration of PSOavg, so that a fixed number of steps represents the same number of function evaluations for both algorithms.

A. Gaussian Distribution

The purpose of the tests with added Gaussian noise is to study the effect of large variances relative to the initial fitness values in the optimization process. We used four increasing

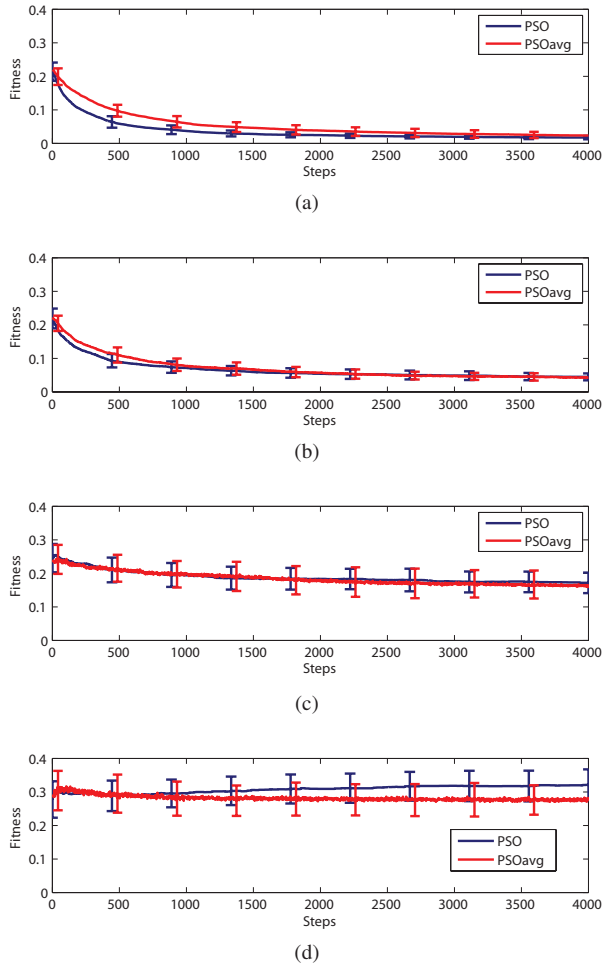


Fig. 5. Fitness of best solution at each step on function f_1 with added Gaussian noise for PSO and PSOavg. A step is equal to one iteration of PSO, and half an iteration of PSOavg. (a) $\sigma = 0$. (b) $\sigma = 0.01$. (c) $\sigma = 0.05$. (d) $\sigma = 0.1$.

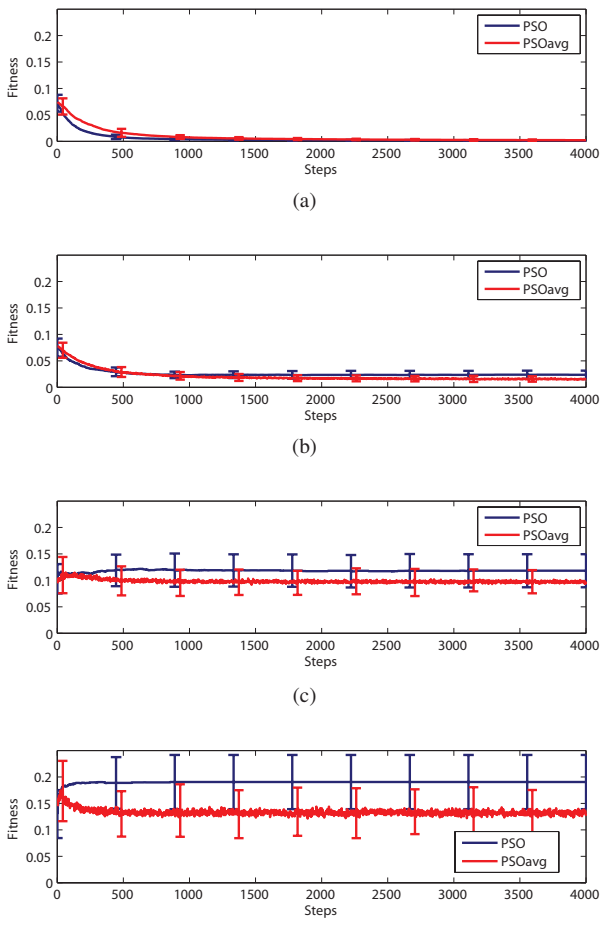


Fig. 6. Fitness of best solution at each step on function f_2 with added Gaussian noise for PSO and PSOavg. A step is equal to one iteration of PSO, and half an iteration of PSOavg. (a) $\sigma = 0$. (b) $\sigma = 0.01$. (c) $\sigma = 0.05$. (d) $\sigma = 0.1$.

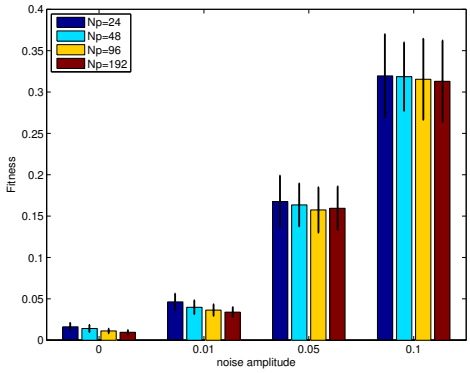
values of the standard deviation: $\sigma = \{0, 0.01, 0.05, 0.1\}$. Figure 5 shows the progress of the learning on benchmark function f_1. For low levels of noise, the algorithm makes progress for a large number of steps. However, for the levels of noise observed in the experiments with robots, the optimization process quickly stagnates, and increasing the number of steps does not improve the performance further.

This effect is not mentioned in previous works on benchmark functions with added noise because the standard deviation values used are much lower than the ones considered in this paper. For example, on the unnormalized sphere function (without dividing by the maximum value as described in the beginning of this section), an unnormalized variance of 1.0 might affect the final stages of the optimization process when the fitness values become small, but is not significant in the initial phases where the values of the function are in the order of $D \cdot x_{init}^2$. However, when the fitness values are normalized to $[0,1]$ and $\sigma = 0.1$, the noise is much more disruptive in the initial stages of the learning. This might explain why the number of iterations used in the robotic learning literature is considerably lower than the ones used on numeric benchmark functions (order of hundreds versus order of thousands).

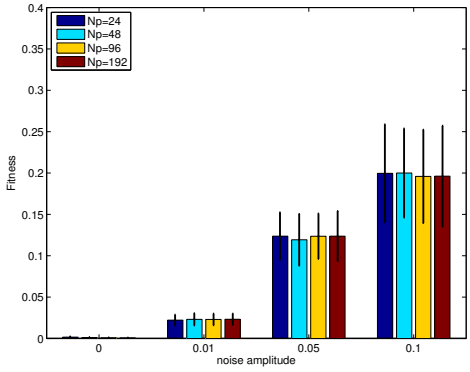
We observe a similar behavior in the normalized benchmark function f_2 (see Figure 6), which suggests that this effect is not particular to an individual fitness function but mainly caused by the amount of noise added.

Under the high-amplitude noise conditions observed in these experiments with $\sigma = 0.05$ and $\sigma = 0.1$, PSOavg significantly outperforms standard PSO.

In the genetic algorithms literature, increasing the population size is often mentioned as an effective technique to deal with noise [4], [5]. In order to test whether this statement also applies to PSO under the high-amplitude noise conditions described previously, we ran standard PSO on benchmark functions f_1 and f_2 with added Gaussian noise and increased the population size from 24 to $\{48, 96, 192\}$ while holding the other parameters constant (i.e., the larger population sizes require a larger number of total function evaluations). Figure 7 shows the final fitness obtained after 4000 iterations and Figure 8 shows the progress on function f_1. It can be seen from Figure 7 that increasing the population size achieves better fitness for low amounts of noise, but it

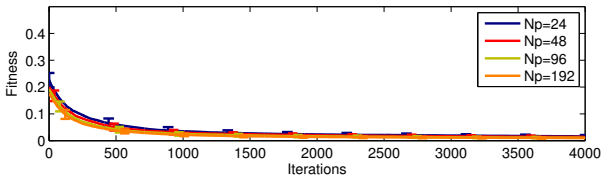


(a)

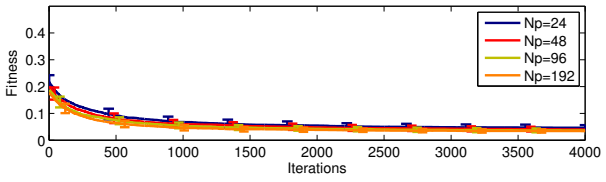


(b)

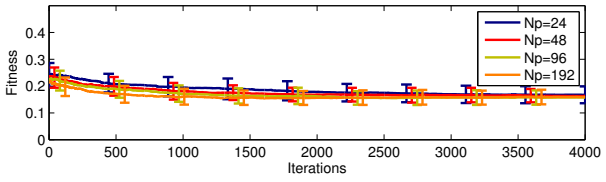
Fig. 7. Final fitness on functions f_1 and f_2 with added Gaussian noise for increasing population sizes. (a) Function f_1. (b) Function f_2.



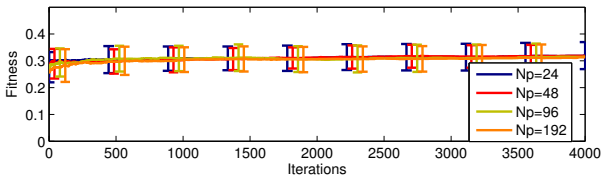
(a)



(b)



(c)



(d)

Fig. 8. Fitness of best solution at each iteration on function f_1 with added Gaussian noise for increasing population sizes. (a) $\sigma = 0$. (b) $\sigma = 0.01$. (c) $\sigma = 0.05$. (d) $\sigma = 0.1$.

does not improve the performance for high-amplitude noise, even though the total number of functions evaluations is much higher. Figure 8 shows that even though the population size was increased eight times the algorithm failed to make progress for $\sigma = 0.1$.

B. Bernoulli Distribution

We employ the Bernoulli distribution to study the effect of skewed noise with positive and negative amplitudes. This is a simplified model of both type of outliers that we observed in robotic learning: bad evaluations of good solutions (e.g., hardware failures), and good evaluations of bad solutions (e.g., unusually advantageous initial conditions).

The variance σ^2 of a Bernoulli distribution with probability p and amplitude A is given by:

$$\sigma^2 = A^2 p(1-p) \tag{6}$$

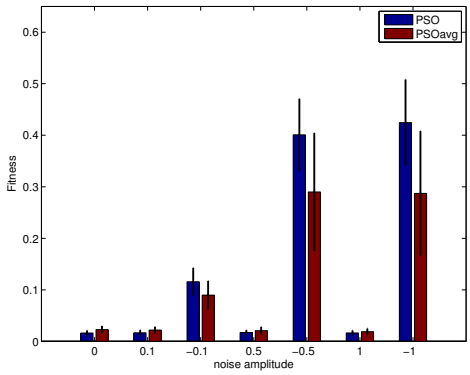
We set $p = 0.01$ and $A = \{0, \pm 0.1, \pm 0.5, \pm 1\}$ in order to obtain the same standard deviation $\sigma = \{0, 0.01, 0.05, 0.1\}$ as used in the experiments with Gaussian Noise. Figure 9 shows the final fitness obtained after 4000 steps on both benchmark functions. In all cases, negative amplitudes perform significantly worse than positive amplitudes of the same magnitude. This means that there is a non-symmetric effect of the noise: good spurious evaluations of bad solutions are worse than bad evaluations of good solutions. PSOavg helps to reduce this effect by discarding bad solutions that had a high fortuitous evaluation through the re-evaluations of the personal best.

Figure 10 shows the progress of the optimization on benchmark function f_1 with added Bernoulli noise of several amplitudes and $p = 0.01$. The algorithm fails to make progress in high-noise settings, as we have shown with the Gaussian noise distribution in both benchmark functions (Figures 5 and 6). We also conducted tests on benchmark function f_2 with added Bernoulli noise and observed the same behavior (graphs not shown).

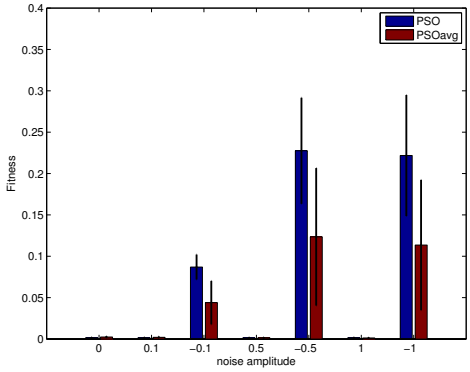
V. Conclusions and Future Work

We have shown that fitness evaluations in multi-robot learning have a large-amplitude noise that is disruptive to the initial phases of the learning process of PSO. We were able to reproduce this behavior on standard benchmark functions by normalizing the fitness values and adding Gaussian noise with a large standard deviation relative to the fitness values obtained at the beginning of the learning process.

We have also modeled two kind of outliers that we observed in multi-robot learning with a Bernoulli distribution using positive and negative amplitudes. We showed that

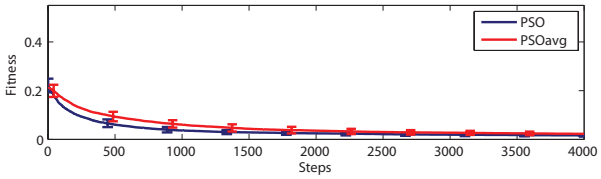


(a)

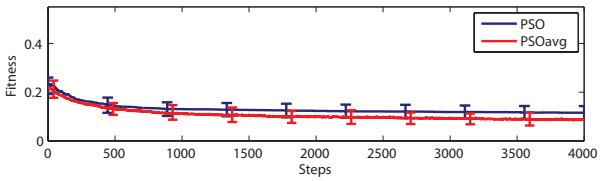


(b)

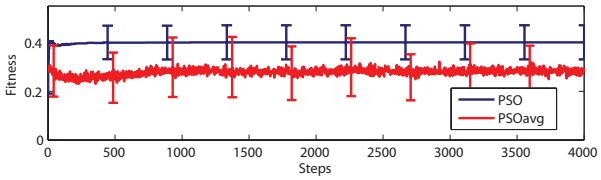
Fig. 9. Final fitness on functions f_1 and f_2 with added Bernoulli noise of positive and negative amplitudes ($p = 0.01$) for PSO and PSOavg. (a) Function f_1. (b) Function f_2.



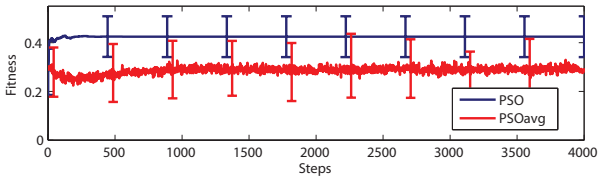
(a)



(b)



(c)



(d)

Fig. 10. Fitness of best solution at each step on function f_1 with added Bernoulli noise ($p = 0.01$) for PSO and PSOavg. A step is equal to one iteration of PSO, and half an iteration of PSOavg. (a) $A = 0$. (b) $A = -0.01$. (c) $A = -0.5$. (d) $A = -1$.

PSO is more sensitive to good spurious evaluations of bad solutions than bad evaluations of good solutions.

Under these conditions, neither increasing the population size nor increasing the number of iterations were able to improve the performance of the learning. On the other hand, we have seen that re-evaluations led to an improvement in performance and are therefore an alternative to deal with noise in multi-robot settings. As part of our future work, we intend to design new targeted strategies for re-evaluating solutions to overcome the challenge of noise in robotic learning.

Acknowledgements

The authors would like to thank Jim Pugh for helpful discussions.

References

[1] Y. Jin and J. Branke, "Evolutionary Optimization in Uncertain Environments: A Survey," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 303–317, June 2005.

[2] K. E. Parsopoulos and M. N. Vrahatis, "Particle Swarm Optimizer in Noisy and Continuously Changing Environments," in *Artificial Intelligence and Soft Computing*, M. H. Hamza, Ed. IASTED/ACTA Press, 2001, pp. 289–294.

[3] H. Pan, L. Wang, and B. Liu, "Particle Swarm Optimization for Function Optimization in Noisy Environment," *Applied Mathematics and Computation*, vol. 181, no. 2, pp. 908–919, Oct. 2006.

[4] J. Fitzpatrick and J. Grefenstette, "Genetic Algorithms in Noisy Environments," *Machine Learning*, vol. 3, no. 2, pp. 101–120, Oct. 1988.

[5] P. Stagge, "Averaging Efficiently in the Presence of Noise," in *Parallel Problem Solving from Nature PPSN V*, ser. Lecture Notes in Computer Science, A. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, Eds. Springer Berlin / Heidelberg, 1998, vol. 1498, pp. 188–197.

[6] D. Arnold and H.-G. Beyer, "A General Noise Model and its Effects on Evolution Strategy Performance," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 380–391, Aug. 2006.

[7] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *IEEE International Conference on Neural Networks*, 1995, pp. 1942 – 1948 vol.4.

[8] J. Hereford and M. Siebold, "Using the Particle Swarm Optimization Algorithm for Robotic Search Applications," in *IEEE Swarm Intelligence Symposium*, 2007, pp. 53–59.

[9] M. Turduev and Y. Atas, "Cooperative Chemical Concentration Map Building Using Decentralized Asynchronous Particle Swarm Optimization Based Search by Mobile Robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 4175–4180.

[10] L. Marques, U. Nunes, and A. T. Almeida, "Particle Swarm-Based Olfactory Guided Search," *Autonomous Robots*, vol. 20, no. 3, pp. 277–287, May 2006.

[11] X. Chen and Y. Li, "Neural Network Predictive Control for a Mobile Robot Using PSO with Controllable Random Exploration Velocity," *International Journal of Intelligent Control and Systems*, vol. 12, no. 3, pp. 217–229, 2007.

[12] R. Poli, "Analysis of the Publications on the Applications of Particle Swarm Optimisation," *Journal of Artificial Evolution and Applications*, vol. 2008, no. 2, pp. 1–10, 2008.

[13] J. Pugh, Y. Zhang, and A. Martinoli, "Particle Swarm Optimization for Unsupervised Robotic Learning," in *IEEE Swarm Intelligence Symposium*, 2005, pp. 92–99.

[14] E. Di Mario, Z. Talebpour, and A. Martinoli, "A Comparison of PSO and Reinforcement Learning for Multi-Robot Obstacle Avoidance," in *IEEE Conference on Evolutionary Computation*, vol. 1, June 2013, pp. 149–156.

[15] E. Di Mario and A. Martinoli, "Distributed Particle Swarm Optimization for Limited Time Adaptation with Real Robots," *Robotica*, vol. 32, no. 02, pp. 193–208, 2014.

[16] D. Floreano and F. Mondada, "Evolution of Homing Navigation in a Real Mobile Robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 3, pp. 396–407, 1996.

[17] J. Pugh and A. Martinoli, "Distributed Scalable Multi-robot Learning Using Particle Swarm Optimization," *Swarm Intelligence*, vol. 3, no. 3, pp. 203–222, May 2009.

[18] B. Huang, G. Cao, and M. Guo, "Reinforcement Learning Neural Network to the Problem of Autonomous Mobile Robot Obstacle Avoidance," in *International Conference on Machine Learning and Cybernetics*, 2005, pp. 85–89.

[19] R. E. Palacios-Leyva, R. Cruz-Alvarez, F. Montes-Gonzalez, and L. Rascon-Perez, "Combination of Reinforcement Learning with Evolution for Automatically Obtaining Robot Neural Controllers," in *IEEE International Conference on Evolutionary Computation*, 2013, pp. 119–126.

[20] E. Di Mario, I. Navarro Oiza, and A. Martinoli, "The Effect of the Environment in the Synthesis of Robotic Controllers: A Case Study in Multi-Robot Obstacle Avoidance using Distributed Particle Swarm Optimization," in *Proceedings of the Twelfth European Conference on the Synthesis and Simulation of Living Systems*. MIT Press, 2013, pp. 561–568.

[21] O. Michel, "Webots: Professional Mobile Robot Simulation," *Advanced Robotic Systems*, vol. 1, no. 1, pp. 39–42, 2004.