SwarmViz: An Open-Source Visualization Tool for Particle Swarm Optimization

Guillaume Jornod, Ezequiel Di Mario, Iñaki Navarro and Alcherio Martinoli
Distributed Intelligent Systems and Algorithms Laboratory
School of Architecture, Civil and Environmental Engineering
École Polytechnique Fédérale de Lausanne
guillaume.jornod@gmail.com, {ezequiel.dimario, inaki.navarro, alcherio.martinoli}@epfl.ch

***Abstract*—Particle Swarm Optimization (PSO) is a metaheuristic for solving high dimensional optimization problems. Due to the large number of dimensions usually employed with PSO, it is not trivial to visualize and monitor the progress of the algorithm. Because of this, adjusting the parameters that govern the dynamics of the swarm for a specific problem becomes challenging. In this article, we present SwarmViz, an open-source visualization tool for PSO. Through SwarmViz, users are able to set up PSO experiments on canonical benchmark functions or input data from external experiments (e.g., learning robotic controllers), and to visualize the optimization process with state-of-the-art visualization tools. SwarmViz has two main goals. First, to enable researchers to monitor the progress of their specific optimization problem and adjust the relevant PSO parameters. Second, to give a visual insight about PSO to students in the scope of teaching optimization techniques. We demonstrate the features of the software through examples on well-known numerical benchmark functions and a case study on the optimization of a robotic controller.**

I. Introduction

This paper introduces SwarmViz, an open-source, cross-platform software written in C++ for the visualization of Particle Swarm Optimization (PSO) [1]. SwarmViz addresses the problem of visualizing the progress of PSO in high-dimensional search spaces for educational and research purposes. The current version of the software is focused on PSO, but it is easily extensible to other population-based algorithms. By its essence, PSO is used to solve multi-dimensional problems. This feature raises an issue when the dynamics of the algorithm are studied: high-dimensional visualization. Even though the visualization of two or three dimensions is straightforward, the addition of extra dimensions involves a large set of additional problems to address.

PSO is taught in higher education institutions in courses related to swarm intelligence and machine learning techniques. This study is generally addressed through the formal definition of the algorithm illustrated by several two-dimensional examples of the swarm dynamics. This educational purpose would largely benefit from a high-dimensional visualization tool.

Moreover, PSO has a significant amount of parameters, which all drastically impact the efficiency of the algorithm. It is therefore of great interest to obtain a better understanding of the influence of such parameters using the proposed visualization tool.

The learning of robotic behaviors with population-based metaheuristic techniques has a large computational cost. The evaluation of each candidate solution leads to a computationally expensive simulation or real robotic experiment, increasing dramatically the cost of the learning. Additionally, multiple sources of uncertainty affect the performance evaluation of a candidate solution and lead to a need of reevaluation [2]. Being able to understand the dynamics of PSO in this kind of application can help in reducing the learning costs.

The presented software enables the visualization of the PSO process using state-of-the-art representation means for population-based algorithms, including projections, representations of the swarm state, and progress metrics.

SwarmViz comes with a set of canonical benchmark functions on which PSO can be tested. Users can set all PSO parameters through a dedicated Graphical User Interface (GUI) and run experiments, either step by step, with a fixed time delay between iterations, or at computational speed. Additionally, Gaussian noise can be added to numerical benchmark function evaluations.

The tool also supports the import of data from external simulated or real experiments. While it is straightforward to store the results of each iteration in text files, their interpretation is difficult when multiple dimensions are involved. This issue is tackled by the possibility to visualize this data through SwarmViz.

The remaining of this article is organized as follows. In Section II, existing visualization tools and techniques are presented together with a quick overview of the PSO algorithm. An overview of SwarmViz and its features is presented in Section III. Section IV presents the different visualizations developed in the tool. The visualization of a robotic learning experiment example is described and discussed in Section V. Finally, Section VI concludes this article and presents an outlook for potential improvements.

II. Related work

A. PSO Algorithm

PSO is a relatively new metaheuristic originally introduced by Kennedy and Eberhart [1], which was inspired by the movement of flocks of birds and schools of fish. Because of its simplicity and versatility, PSO has been used in a wide range of applications such as antenna design, communication

```
Intialize particles
for N_i iterations do
    for each particle do
        Update particle position
        Evaluate particle
        Share personal best
    end for
end for
```

Fig. 1. PSO algorithm.

networks, finance, power systems, and scheduling. The pseudocode for the algorithm is shown in Figure 1.

Given $x_{i,j}$ the position of particle i in dimension j, its velocity $v_{i,j}$ 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.

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

B. Visualization of High-Dimensional Optimization Problems

Spears presented an overview of multidimensional visualization [3]. This overview gives insights on how to use colors in the visualization of evolutionary algorithms. The idea of depicting each generation with a different color can be extended to PSO in order to differentiate particles and iterations. In addition to differentiating multiple objects on the same plot, colors can be used to denote an additional dimension with a color map.

This overview also presents glyphs, i.e., elements of writing or complex symbols, such as the star plot and *Chernoff's faces*. The former represents the dimensions with star spokes whose lengths denote the values in a relative scale. The latter represents the data point with traits on human faces. The implementation of *Chernoff's faces* is obviously very greedy, especially when it comes to representing the states of the algorithm with no delay. The different representation methods using glyphs are reviewed in another study [4].

In addition, Spears reviews projection techniques [3]. He states that the simple projection in 2D of multidimensional data is of great interest since it is computationally simple and might reveal unusual structures within the multidimensional space. *Andrew's curves* [5] and the *Grand tour* [6] techniques are also presented.

Finally, the overview ends with the parallel coordinates plot, a method that allows to work around one drawback of the projections: the impossibility to visualize all dimensions in the same plot. The multidimensional points are drawn on a broken line in which each vertex corresponds to a dimension.

A more recent overview of multidimensional visualizations focused on PSO presents results with the density plot, a graph that shows whether the particles converged, and the parallel coordinates plot, described above, which can help to detect convergence or specific patterns [7]. These two visualization techniques are described in further detail in Sec. IV.

C. Existing Tools

GEATbx [8] is a MATLAB toolbox that provides graphs to visualize the convergence of Genetic Algorithms. It includes the visualization of the fitness of individuals, distance between them and other metrics. The tool offers complete sources and documentation; however, it is a commercial software with a large cost which can be a potential drawback for educational purposes.

GONZO [9] is a software visualization tool for Evolutionary Algorithms. It offers the possibility to display summary graphs with genetic and parental information, as well as search space metrics. This tool works for online and offline simulations, which allows to visualize ongoing and finished simulations.

VISPLORE [10] is a toolkit for PSO exploration which runs on Mathematica. It includes density plots, star plots, parallel coordinates plots, and history plots, and offers as well the possibility to visualize a single particle, the swarm at a specific step, the swarm during an entire experiment or multiple experiments.

The aforementioned tools offer a large set of possibilities: multiple plotting and visualization techniques, visualization of different algorithms and running on different platforms with both commercial and non-commercial software.

However, the research and educational purposes addressed in this paper require a high flexibility for the former and a low or non existent cost for the latter. Moreover, the flexibility should be extensible to both platforms and algorithms.

The development of a standalone application has the advantage of being independent of any license. Furthermore, it gives freedom for users to customize the algorithm being used, adapting it to any of the PSO variants present in the literature, and the benchmark functions, for instance, by adding noise with different distributions. It also provides the possibility to load external data. These data can originate from either simulated or real-life experiments, and the input format can be adapted to the researcher's needs.

III. Software Overview

This section provides an overview of SwarmViz features, including the different visualizations, the capabilities for interpretation of data generated offline, and the execution of PSO experiments on canonical benchmark functions.

SwarmViz was developed in the C++ language using Qt, an application framework which enables the development of cross-platform graphical user interfaces. Benefiting from the cross-platform advantage of Qt applications, the tool has been successfully tested on multiple operating systems: Ubuntu 12.04, Linux Mint 17.1, Windows 7, and OS X 10.9. This application is distributed under the GNU General Public License. Its source code is available in the git repository `https://github.com/epfl-disal/SwarmViz.git`.

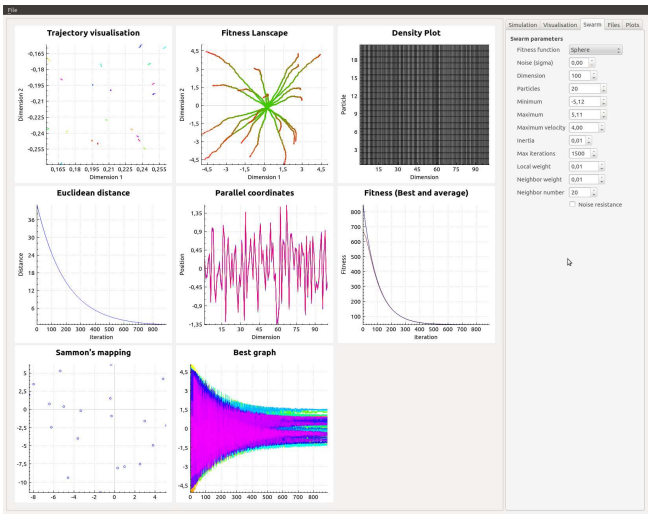


Fig. 2. *SwarmViz* main window with the different visualizations on the left, and the configuration and control panel tabs on the right side.

A. A GUI for PSO Experiment Visualizations

From a development point of view, a visualization tool could a priori only be composed of independent plot windows launched from a command line. But the tasks to be carried out are multiple and diverse: tweaking PSO parameters, reading and writing files, tuning visualization options, etc. Consequently, SwarmViz is under the form of a GUI designed to setup simulations, manage input and output files and handle the visualization widgets. Figure 2 presents the main window of the software.

The two main usages of the software are the analysis of PSO on benchmark functions and visualization of off-line PSO experiments, in our case from robotic learning. The former is tackled through the configuration of PSO experiments with the desired parameters using the GUI, which is discussed in Sec. III-B, while the latter is addressed by the input and reading of data generated offline, which is detailed in Sec. III-C. Both functions of the software are useful for research and education.

B. PSO on Benchmark Functions

The configuration of a PSO experiment begins with the choice of the function among the set of implemented canonical functions presented in Table I. Then, the algorithm parameters are specified:

- number of particles;
- dimensionality of the problem;
- minimum initial value;
- maximum initial value;
- maximum velocity;
- total number of neighbors;
- weight of the neighborhood best;
- weight of the personal best;
- inertia;
- maximum number of iterations;
- number of re-evaluations.

By setting a non zero value for the inertia parameter, PSO with inertia is run [11]. Moreover, a noise value can be defined; this value is the standard deviation of the Gaussian noise that will be added to the function evaluations. Finally, a noise-resistant version of the algorithm [12], [13] can be activated.

The software also includes the option to write the state of the algorithm on benchmark functions in the same format as the one used to load external data, intended to be used for reproducing results of previous experiments.

C. Loading data generated offline

In the process of robotic learning with PSO, it is common to output the states of the particles at each iteration. Analyzing information from this kind of data is difficult, especially with large swarms on high dimensional problems and long learning times.

SwarmViz proposes a feature to read data generated offline. This feature lets the user import files from robotic experiments, allowing to visualize the behavior of the swarm during the learning process and to draw conclusions on the influence of the PSO parameters in the scope of robotic learning.

D. Ouputs

In addition to the aforementioned state files, SwarmViz offers the possibility to output the plots at any time of the experiment. The available formats are PDF, JPG, PNG and BMP, which come with specific properties: size, quality and scale. These figures can be directly included in reports and articles as illustrated in the examples of this paper.

IV. Visualizations

This section presents the visualizations provided by SwarmViz. The term visualization as used in this section should be understood as graphical representation. The visualizations described are: trajectories, fitness landscape, Sammon's mapping, density, parallel coordinates, best graph, Euclidean distance, and finally fitness. They are classified by types: projections, overall visualizations, and metrics.

A. Projections

Projections can be used to reduce the number of dimensions of the original data to a number suitable for graphical representation, i.e., two or three dimensions. Three projections implemented in the visualization tool are presented.

1) Trajectories: they depict the change in the particles' positions at successive iterations. The visualization of trajectories is achieved through a projection from the search space to a plane of two arbitrary dimensions selected by the user. Two coloring schemes are proposed, the first one is index-based whereas the second one is based on the performance of the particles at the current iteration.

Figure 3 illustrates trajectories with a swarm of 20 particles on the sphere function in 100 dimensions, using index-based colors. In this example, the inertia, and the neighborhood and local weights are very low, which allows to draw simultaneously fifty positions of a particle and to get a *worms* effect. The

TABLE I
LIST OF IMPLEMENTED CANONICAL FUNCTIONS.

Name	Definition	Domain	Global minimum
Sphere function	$f(x) = \sum_{j=1}^{d} x_j^2$	$\mathbb{R}^d$	$\min(f) = f(\vec{0}) = 0$
Rosenbrock's valley	$f(x) = \sum_{j=1}^{d-1} \left[100 \cdot (x_{j+1} - x_j^2)^2 + (1 - x_j)^2\right]$	$\mathbb{R}^d$	$\min(f) = 0$
Rastrigin's function	$f(x) = 10 \cdot d + \sum_{j=1}^{d} \left[x_j^2 - 10 \cdot \cos(2 \cdot \pi \cdot x_j)\right]$	$\mathbb{R}^d$	$\min(f) = f(\vec{0}) = 0$
Griewank's function	$f(x) = 1 + \sum_{j=1}^{d} \frac{x_j^2}{4000} - \prod_{j=1}^{s} \cos\left(\frac{x_j}{\sqrt{j}}\right)$	$\mathbb{R}^d$	$\min(f) = f(\vec{0}) = 0$
Twin peaks function	$f(x_1, x_2) = -5 \cdot x_1^2 + x_2^2 + x_1^4$	$\mathbb{R}^2$	$\min(f) = f(\pm 1.5811, 0) = -6.25$
Crater lake function	$f(x_1, x_2) = -\left(x_1^2 + x_2^2\right)^2 + 100 \cdot \left(x_1^2 + x_2^2\right)$	$\mathbb{R}^2$	–
Schwefel's function	$f(x) = \sum_{j=1}^{d} \left[-x_j \cdot \sin\left(\sqrt{\lvert x_j \rvert}\right)\right]$	$\mathbb{R}^d$	$\min(f) = f(420.\vec{9}687) = -418.9829 \cdot d$
Ackley's function	$f(x) = -20 \cdot \exp\left[-0.2 \cdot \sqrt{\frac{1}{n} \sum_{j=1}^{d} x_j^2} - \exp\left(\frac{1}{n} \sum_{j=1}^{d} \cos 2 \cdot \pi \cdot x_j\right)\right] + 20 + \exp(1)$	$\mathbb{R}^d$	$\min(f) = f(\vec{0}) = 0$
Michalewicz's function	$f(x) = -\sum_{j=1}^{m} \sin(x_j) \cdot \left[\sin\left(\frac{j \cdot x_j^2}{\pi}\right)\right]^{20}$	$\mathbb{R}^d$	–
Easom's function	$f(x_1, x_2) = -\cos(x_1) \cdot \cos(x_2) \cdot \exp\left(-(x_1 - \pi)^2 - (x_1 - \pi)^2\right)$	$\mathbb{R}^2$	$\min(f) = f(\pi, \pi) = -1$
Drop wave function	$f(x_1, x_2) = -\frac{1 + \cos\left(12 \cdot \sqrt{x_1^2 + x_2^2}\right)}{0.5 \cdot (x_1^2 + x_2^2) + 2}$	$\mathbb{R}^2$	–

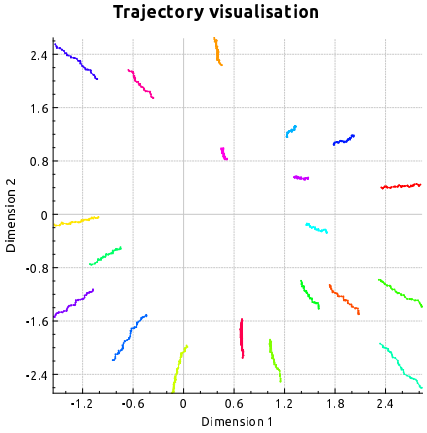
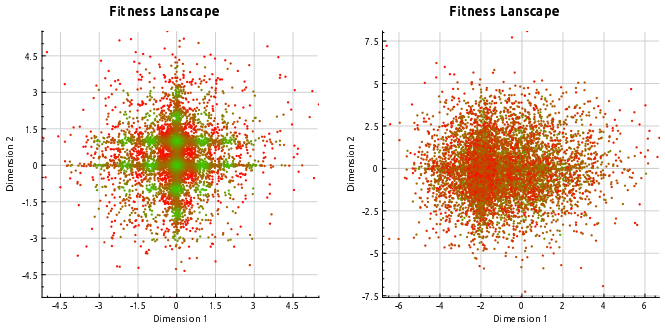


Fig. 3. Trajectory visualization on the sphere function in 100 dimensions with 20 particles and 50 successive positions for each trajectory.



(a) (b)

Fig. 4. Fitness landscape for the Rastrigin function (20 particles after 400 iterations). (a) Two dimensions. (b) Ten dimensions.

particles slowly move towards the global minimum, located at the origin.

2) Fitness Landscape: it describes the attempt to reconstruct the shape of the fitness function using all previously evaluated positions. Each position is projected into a two-dimensional plane using a color coding to represent the fitness function values. This plot is similar to the visualization of trajectories, except that all positions are represented in the selected dimensions, with no identification of the different particles. As it is often the case when projections are used, this visualization has the drawback that positions that are far apart in the original search space may actually overlap in the projected space, causing clutter and potentially leading to confusion. However, it can also provide insight as to whether there is some kind of structure in the search space and help to identify or discard potential regions for good solutions.

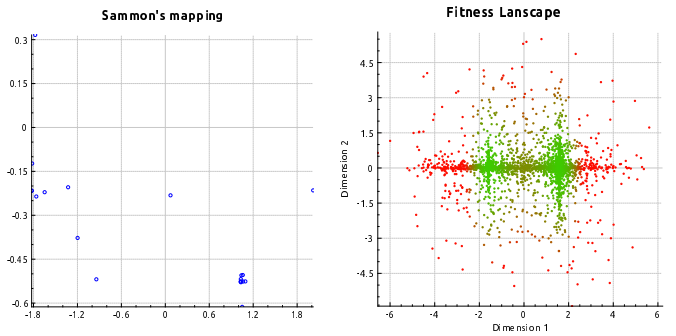
Figure 4 illustrates the fitness landscape visualization for a simulation with 20 particles in 400 iterations, in two and ten dimensions on the Rastrigin function. In two dimensions, the Rastrigin fitness landscape is expected to show green circles, depicting the local minima of the function. However, in ten dimensions, this pattern is less clearly visible, principally due to the overlapping of different performances at the same point.

3) Sammon's mapping: it is a non-linear algorithm that maps a high dimensional space to a lower dimensional space (i.e., dimension reduction), while attempting to preserve the distances between points [14]. This preservation is obtained by minimizing a metric, called Sammon's stress:

$$E = \frac{1}{\sum_{i<j} \delta_{ij}} \sum_{i<j} \frac{(\delta_{ij} - d_{ij})^2}{\delta_{ij}}$$

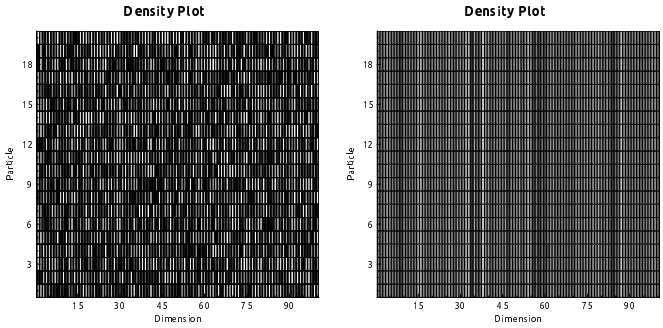
where δ_{ij} and d_{ij} denote the distance in the d-dimensional space and in the resulting 2-dimensional space, respectively.

The minimization of the Sammon's error is performed by gradient descent. The gradient, the Hessian matrix, and the search direction are recomputed until the error is lower than 10^{-6}.



(a) (b)

Fig. 5. Clustering of the particles in the Twin Peaks function in two dimensions for a swarm of 20 particles. (a) Sammon's Mapping. (b) Fitness landscape.



(a) (b)

Fig. 6. Density plots for sphere function with 20 particles and 100 dimensions. (a) Plot at the initial state. (b) Plot at the 75th iteration.

The current implementation does not take into account the previous state of the particles, which leads to abrupt transitions between iterations in the projected space, making it hard to follow the particles. This issue could be addressed by implementing a variation of Sammon's mapping that would take into account the previous swarm state and allow to visualize the trajectories in the projected space, as described in [15].

Figure 5 shows a simple case of pattern observation in two dimensions. The purpose of this example is not to show the dimensionality reduction but to illustrate the cluster detection. The twin peaks function in two dimensions presents two global optima (Fig. 5b); the particles fall into one of the two minima (Fig. 5a). Figure 5 also highlights the orientation loss: the plane of the Sammon's mapping is not related to the orientation of the axes in the original plane.

B. Overall visualizations

Trajectories and fitness landscape visualizations let us observe the particles in a two dimensional projection, but the behavior in the other dimensions is hidden. This could become an issue when we are trying to detect the convergence of the algorithm. Sammon's mapping is a good attempt to work around this drawback, but the actual values and the scale differences are lost.

SwarmViz implements three additional visualizations that offer an overview of the positions in the d-dimensional space, which we name overall visualizations.

1) Density: it is a two dimensional graphical tool originally used to visualize data contained in a matrix. It is widely used in the field of System Biology to visualize DNA microarray data. This plot is also known under the previously trademarked name of *Heatmap*.

Here, the density plot is used to visualize all the dimensions of the PSO problem at a specific iteration. Each rectangle represents the position of a particle in a dimension. For instance, the rectangle at (1, 10) represents the position of the first particle in the tenth dimension. The intensity of the gray color in the rectangle denotes the value of that position, using one of several color scales proposed (e.g., black for the lowest position value found so far, white for the highest). In addition to giving an overview of the values of the particles, this plot aims to detect the convergence of the algorithm, depicted by vertical lines which represent the same position for all particles in a given dimension.

Figure 6 shows the density plots for PSO with 20 particles on the sphere function in 100 dimensions, at the 1st and 75th iterations. The random values for the initial state clearly appear on Fig. 6a, depicted by a gray scale patchwork. The relative convergence of the swarm at the 75th iteration is depicted by vertical lines on Fig. 6b.

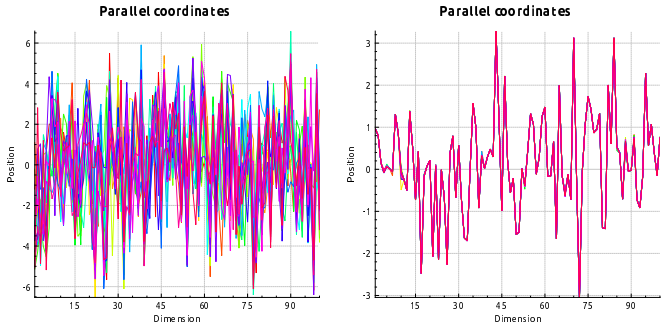
2) Parallel Coordinates: it is a multi-dimensional visualization tool. This plot is largely used in different domains such as data mining, air traffic control, and computer vision.

The particles are represented by poly-lines; their vertices are located at points whose abscissas represent dimensions and ordinates their corresponding values. In addition to its original purpose, i.e., an overview of the particles' positions in all dimensions, the parallel coordinates plot is also useful to detect convergence: overlapping vertices represent equal values in the corresponding dimension and overlapping poly-lines show convergence of the algorithm.

Figure 7 shows the parallel coordinates plot in the same experiment than the density plot in the previous section. The initial values shown in Fig. 7a represent the randomness of the particle initialization as in Fig. 6a for the density plot. In addition to providing a way to detect convergence, depicted by the overlapping polylines, Fig. 7b gives an overview of the optimized values.

3) Best graph: This graph reports the position of the best particle across all iterations. This visualization is an adaptation of the depiction of the *Best individual* [16] used in Genetic Algorithms.

Each dimension of the global best is represented by a specifically colored line, whose vertices denote a particular position value for that dimension at each iteration. It should be noted that the best particle is the particle with the best estimated performance, which is not necessarily the actual best in the case of noisy function evaluations.



(a) (b)

Fig. 7. Parallel coordinates plot for the sphere function in 100 dimensions with 20 particles. (a) At the first iteration. (b) At the 75^{th} iteration.

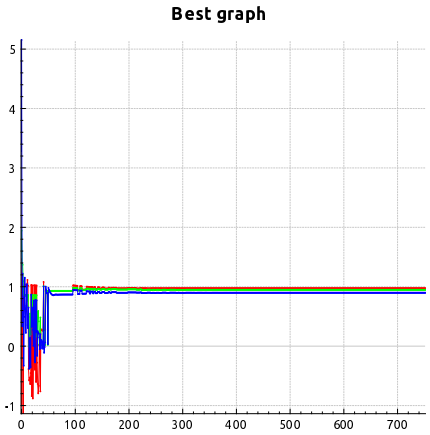


Fig. 8. Best graph plot for a swarm of 20 particles on the Rastrigin function in 3 dimensions with a global neighborhood.

This visualization is useful to observe the convergence of the algorithm, i.e., when the global best stabilizes in a certain position. The best particle is also a good indicator of the state of the algorithm, as it guides the search for the other particles, especially in the case of a global neighborhood with a large weight w_n in Eq. 1.

Figure 8 illustrates the influence of the neighborhood weight in the case of 20 particles in an experiment on the Rastrigin function with 3 dimensions. The best particle moves during the first 50 iterations to finally find a stable position. A sign of convergence is observed after the iteration 120, where the three dimensions are close to 1. In this case, the location $[1,1,1]$ is the best local minimum.

C. Metrics

This section presents two metrics that describe the progress of the algorithm through different iterations.

1) Euclidean Distance: it is computed as the mean inter-particle distance at each iteration for all particles in the swarm:

$$d_e = \frac{1}{n(n-1)} \cdot \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2} \qquad (2)$$

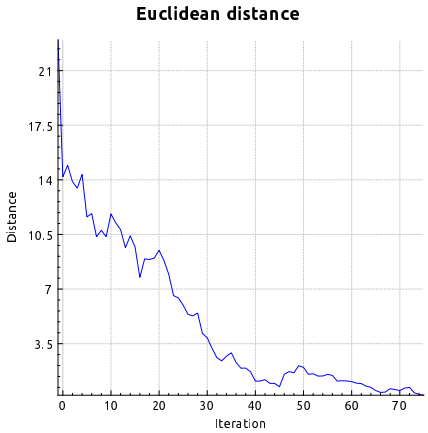


Fig. 9. Euclidean distance plot for a swarm of 20 particles on the sphere function in 100 dimensions during 75 iterations.

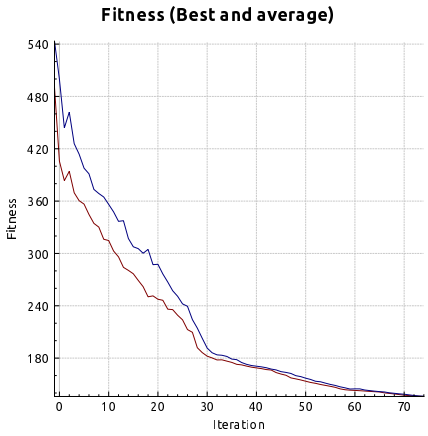


Fig. 10. Fitness plot for a swarm of 20 particles on the sphere function in 100 dimensions during 75 iterations.

As shown in Fig. 9, it gives an overall measure of the compactness of the swarm and of convergence towards a particular position.

2) Fitness: The learning process consists of the minimization or the maximization of a fitness function, which is used to evaluate the particles in the swarm. The fitness visualization (see Fig. 10) is a plot of the fitness value as a function of the iterations. The user can select to plot either the fitness of the best particle and the mean of the whole swarm, or the fitness of all individual particles.

V. Case Study

This case study describes the use of the software during the final project of our master-level course on Distributed Intelligent Systems[1]. Students used SwarmViz in order to better understand the dynamics of PSO applied to the learning of robotic controllers. PSO was used to learn a behavior consisting on the formation of a chain constituted by four e-puck robots [17] without communication. The leader had a

[1] http://disal.epfl.ch/teaching/distributed_intelligent_systems

predefined controller, while the three followers ran the same learned controller. This controller was a recurrent artificial neural network of two units with sigmoidal activation functions. The outputs of the units determined the wheel speeds. By exploiting the symmetry of the problem, the total number of parameters to be learned was reduced to five.

The learning problem for PSO was choosing a set of parameters of the underlying robotic controller such that a given fitness metric was minimized. The fitness corresponded to the average of the distance between each pair of robots, which reflected the formation of a robotic chain. The experiments were performed in simulation using Webots [18], a high-fidelity robotic simulator. The PSO swarm consisted of 30 particles, and learning was performed during 93 iterations.

At each iteration, the learning algorithm outputs a file summarizing the state of the swarm, including the particles' position, velocity, neighbor's and personal best and performance, as well as the best particle of the swarm and its performance. Using these files, the learning process was monitored through SwarmViz. Figure 11 presents the visualizations of the last iteration.

The fitness landscape (Fig. 11a) shows a red cluster (bad performances) at the convergence value ($x_1 \approx -2$, $x_2 \approx -5.5$). This illustrates the difficulty of the algorithm to converge in other dimensions.

Both the parallel coordinates and density plots (Fig. 11b and 11c respectively) show the relative convergence of the algorithm. Dimensions 1, 2 and 4 seem to have a consensus for the convergence values. However, the particles are less clustered in dimensions 3 and 5, emphasizing either the sensitivity of these dimensions or the difficulty to find an optimal value. The best graph (Fig. 11d) confirms these trends. The patterns imply that the neighborhood coefficient could be lowered to avoid the high variation of the best position.

The Euclidean distance plot (Fig. 11e) also illustrates the convergence process. The fluctuating decrease of the distance is usual in robotic learning but sharp changes such as around the 16^{th} iteration imply that the PSO parameters are not optimally set. Finally, the average performance (Fig. 11f) is relatively low from the beginning, showing that a quite satisfactory value is found at the early stage of the algorithm.

Students reported that using SwarmViz helped in the choice of the PSO parameters and decreased the number of attempts before reaching a good performing controller. Moreover, they included output graphs in their report and presentation, illustrating the PSO learning and the quantitative description of the results.

VI. Conclusion and Outlook

This paper has described the development of SwarmViz, a visualization tool for PSO. SwarmViz is an open-source software written in C++ aiming to be used as a tool for teaching and research in population-based learning techniques.

SwarmViz has been used in robotic learning and on benchmark functions with added noise both for research and educational purposes. It allowed both researchers and students to better understand the dynamics of PSO. With the publication of the source code we hope to allow other people to benefit from this tool and also to potentially collaborate on augmenting its functionality.

The current version of the software gives a satisfactory visualization of the algorithm's behavior. Nevertheless, we will now discuss features that could be improved in future releases.

The Density Plot implemented in the current version of the software displays the positions of the particles but does not give any information about the performance. On the other hand, the Fitness Plot shows the performance of all particles but due to the fact that it shows all iterations, the performances are not distinguishable at the end of the algorithm. A heatmap such as the one described in [19] combines the values in the search space and the performances; this multi-objective implementation could be adapted for the purpose of this application.

The concrete result of a PSO experiment is the set of values that gives the best performance. In the current state, these positions are only visible in the parallel coordinates plot if the algorithm converges at the end of the experiment. This optimal set of values could be displayed in a more convenient way.

The visualization tool developed only includes PSO as optimization algorithm. Given the structure of the implementation, and the visualizations provided, other population-based algorithms such as GA or Evolution Strategies could be included in the software. A particular development effort has been put to enable this possible extension.

Acknowledgment

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

References

[1] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *IEEE International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.

[2] E. Di Mario, I. Navarro, and A. Martinoli, "Analysis of fitness noise in particle swarm optimization: From robotic learning to benchmark functions," in *IEEE Congress on Evolutionary Computation*, 2014, pp. 2785–2792.

[3] W. M. Spears, "An overview of multidimensional visualization techniques," in *Visualization Workshop of GECCO*, vol. 99, 1999.

[4] S. E. Fienberg, "Graphical methods in statistics," *The American Statistician*, vol. 33, no. 4, pp. 165–178, 1979.

[5] D. F. Andrews, "Plots of high-dimensional data," *Biometrics*, pp. 125–136, 1972.

[6] D. Asimov, "The grand tour: a tool for viewing multidimensional data," *SIAM Journal on Scientific and Statistical Computing*, vol. 6, no. 1, pp. 128–143, 1985.

[7] N. Khemka and C. Jacob, "What hides in dimension x? a quest for visualizing particle swarms," in *International Conference on Ant Colony Optimization and Swarm Intelligence*, 2008, pp. 191–202.

[8] H. Pohlheim, "Geatbx: Genetic and evolutionary algorithm toolbox for use with matlab," *IEE Colloquium on Applied Control Techniques Using MATLAB*, vol. 14, no. 1, 1998.

[9] T. D. Collins, "Understanding evolutionary computing: A hands on approach," in *IEEE Congress on Evolutionary Computation*, 1998, pp. 564–569.

[10] N. Khemka and C. Jacob, "Visplore: a toolkit to explore particle swarms by visual inspection," in *Conference on Genetic and Evolutionary Computation*, 2009, pp. 41–48.

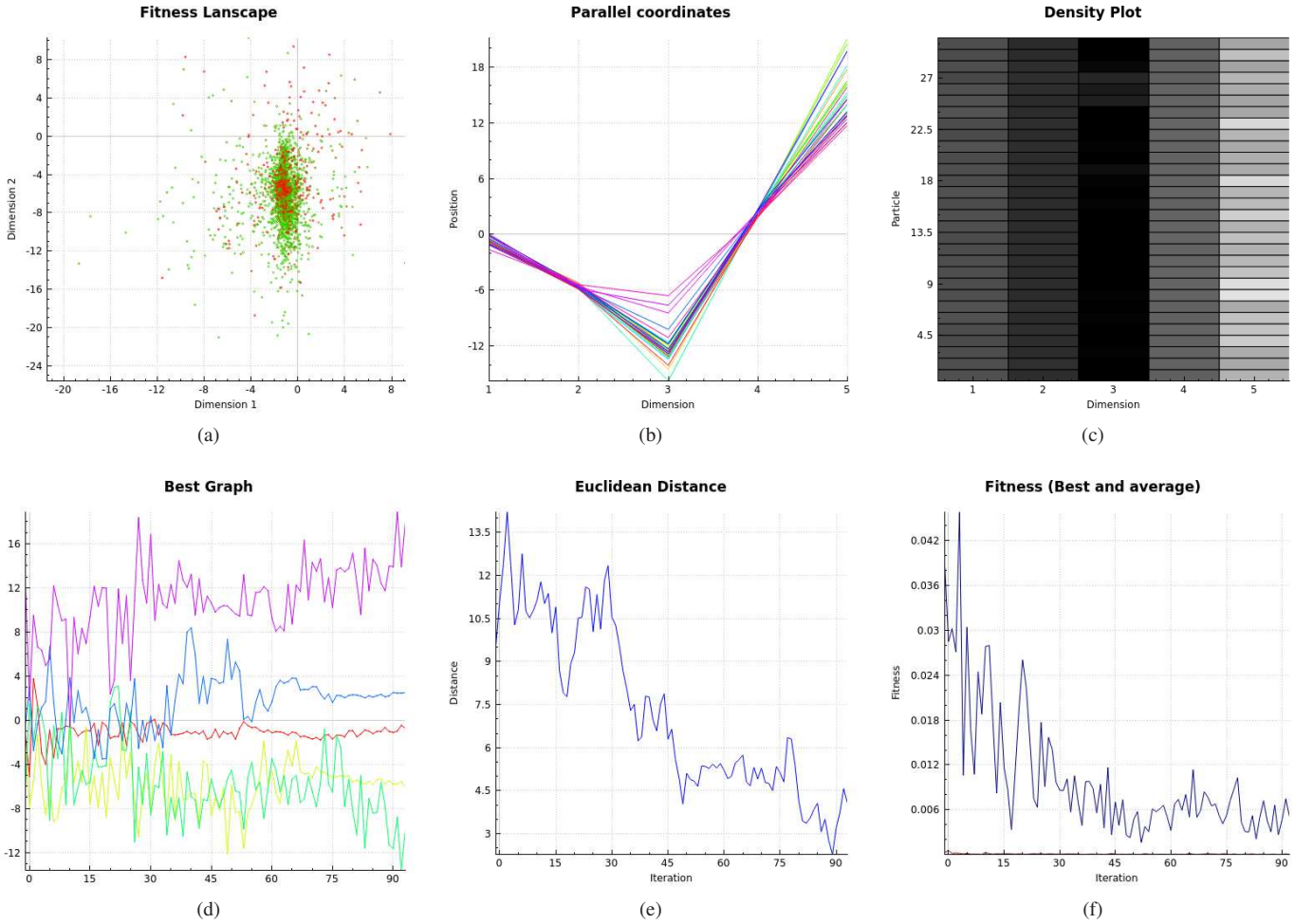


Fig. 11. Final iteration of the robotic learning process. (a) Fitness landscape in the two first dimensions. (b) Parallel coordinates. (c) Density plot with global values scaling. (d) Best graph. (e) Euclidean distances. (f) Best (blue) and average (brown) performances.

[11] R. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," in *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, vol. 1, 2001, pp. 81–86 vol. 1.

[12] J. Pugh, A. Martinoli, and Y. Zhang, "Particle swarm optimization for unsupervised robotic learning," in *IEEE Swarm Intelligence Symposium*, 2005, pp. 92–99.

[13] 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.

[14] J. W. Sammon Jr, "A nonlinear mapping for data structure analysis," *Computers, IEEE Transactions on*, vol. 100, no. 5, pp. 401–409, 1969.

[15] Y.-H. Kim, K. H. Lee, and Y. Yoon, "Visualizing the search process of particle swarm optimization," in *Conference on Genetic and Evolutionary Computation*, 2009, pp. 49–56.

[16] H. Pohlheim, "Visualization of evolutionary algorithms-set of standard techniques and multidimensional visualization," in *Conference on Genetic and Evolutionary Computation*, vol. 1, 1999, pp. 533–540.

[17] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli, "The e-puck, a robot designed for education in engineering," in *Conference on Autonomous Robot Systems and Competitions*, vol. 1, no. 1, 2009, pp. 59–65.

[18] O. Michel, "Webots: Professional mobile robot simulation," *Advanced Robotic Systems*, vol. 1, no. 1, pp. 39–42, 2004.

[19] A. Pryke, S. Mostaghim, and A. Nazemi, "Heatmap visualization of population based multi objective algorithms," in *Evolutionary Multi-Criterion Optimization*. Springer, 2007, pp. 361–375.