Exploratory Toolkit for Evolutionary And Swarm−based Optimization

Namrata Khemka, Christian Jacob

University of Calgary, Calgary, AB, Canada, T2N1N4
{khemka, jacob}@cpsc.ucalgary.ca

Optimization of parameters or 'systems' in general plays an ever−increasing role in mathematics, economics, engineering, and life sciences. As a result, a wide variety of both traditional analytical, mathematical and non−traditional algorithmic approaches have been introduced to solve challenging and practically relevant optimization problems.

Evolutionary optimization methods—namely, *genetic algorithms, genetic programming,* and *evolution strategies*—represent a category of non−traditional optimization algorithms drawing inspirations from the process of natural evolution. *Particle swarm optimization* represents another set of more recently developed algorithmic optimizers inspired by social behaviours of organisms such as birds [8] and social insects. These new evolutionary approaches in optimization are now entering the stage, and are thus far very successful in solving real−world optimization problems [12].

Although these evolutionary approaches share many concepts, each one has its strengths and weaknesses. The best way to understand these techniques is through practical experience, in particular on smaller−scale problems or on commonly accepted benchmark functions. In [11], we describe how *evolution strategies* and *particle swarm optimizers* compare on benchmarks prepared for a much more complex optimization task regarding a kinematic model of a soccer kick. The *Mathematica* notebooks that we created throughout these evaluation experiments and for the final design of the muscle control algorithms for the soccer kick are now also available through a webMathematica interface.

The new *Evolutionary & Swarm Optimization* web site is integrated with the collection of notebooks from the EVOLVICA package, which covers evolution−based optimizers from genetic algorithms and evolution strategies to evolutionary programming and genetic programming. The EVOLVICA database of notebooks, along with the newly added swarm algorithms, provide a large experimentation and inquiry platform for introducing evolutionary and swarm−based optimization techniques to those who either wish to further their knowledge in the evolutionary computation domain or require a streamlined platform to build prototypical strategies to solve their optimization tasks. Making these notebooks available through a web*Mathematica* site means that anyone with an internet browser available will have instant access to a wide range of optimization algorithms.

■ 1. Introduction

Evolutionary optimization methods—*genetic algorithms* (GA) [7], *genetic programming* (GP) [13], and *evolution strategies* (ES) [14]—are a branch of non−traditional optimization methods drawing inspirations from the processes of natural evolution. *Particle swarm optimizers* (PSO), on the other hand, are inspired by the social behaviour of bird flocking [10]. Over the last two years, we have been investigating the performance of evolution− and swarm−based optimizers in the domain of biomechanics, which we developed together with the Human Performance Laboratory at the Faculty of Kinesiology, University of Calgary [1],[2],[3]. In this particular biomechanical application, numerical optimization algorithms are used to design equipment for sports activities. The involved simulations of muscle movements are very time−consuming and high−dimensional, thus making their evaluations costly and difficult. Simulating a soccer kick is an example of such a model which investigates muscle activation patterns within the leg and foot when kicking a soccer ball towards the goal. The specific objective in this case is to obtain a high ball speed; in order to minimzie the goal keeper's chances to catch the ball. In 1998, Cole applied a (1+ λ) ES to this model [1],[2]. More recently we presented improved adaptations of the model parameters through a particle swarm optimizer [11][12].

The main focus of this paper, however, is not on the optimization of parameters for the investigated soccer kick model. Instead, we present what we have learnt from our comparison studies of the evolution− and swarm−based optimizers on a set of selected benchmark functions. These benchmark studies turned out to be extremely useful in understanding the intricacies in performance regarding three optimizers: (1) the originally used (1+ λ) ES, (2) a canonical ('basic') PSO (bPSO), and a (3) a particle swarm optimizer with noise−induced ('random') inertia weight settings (rPSO). We will describe and analyze the performance of each of these optimizers on five benchmark functions in two, four, and ten dimensions and try to project these findings to performance characteristics we have also found in the real−world application of the discussed soccer kick model, which poses a 56−dimensional optimization problem. The *Mathematica* notebooks that we created, provide us with insights regarding the relations between control parameteres and system performance of optimizers under study. Consequently, we gain a better understanding of the algorithms on multi−dimensional real−world problems.

This paper is organized as follows. In Section 2, we give descriptions of the three optimization algorithms used in our comparision. An introduction of the benchmark functions and an outline of the experimental setups follows in Section 3 and Section 4, respectively. We discuss the experimental results, and summarize our lessons learnt in Section 5. The accompanying web*Mathematica* site is presented in Section 6. Section 7 concludes the paper.

■ 2. The Three Contenders: ES, bPSO, and rPSO

The three contenders for our comparative study of evolution− and swarm−based optimization algorithms are: (1) a relatively simple (1+λ) Evolution Strategy, (2) a canonical ('basic') PSO (bPSO), and (3) a particle swarm optimizer with noise−induced ('random') inertia weight settings (rPSO). The following subsections present these approaches in more detail.

□ 1. (1 + λ) Evolution Strategies: ES

Evolution Strategies (ES) have been a successful evolutionary technique for solving complex optimization problems since the 1960's [14]. ES evolves vectors of real numbers and the 'genetic' information is interchanged between these vectors through recombination operators. Slight variations ('mutations') on these vectors are obtained by evolving strategy parameters which determine the amount of mutations applied to each vector component.

As stated above, the original experiments on the soccer kick simulation were conducted by Cole using a (1 + λ) ES scheme [1], [2], [3]. As these experiments were difficult to repeat, we also used this version for our comparison test. In the (1 + λ) ES scheme, a single parent is mutated λ times. Each of the newly created offspring are evaluated and the parents and the offspring are added to the selection pool. The single best individual among the 1 + λ solutions in the pool survives and becomes the parent for the next iteration. Below we describe the $(\mu/\rho + \lambda)$ strategy which is a generalization of the (1 + λ) scheme, where μ parents generate λ offspring with recombination of ρ individuals.

(μ/ρ + λ) ES algorithm

- **Step 1:** Initialize the population of size n by randomly assigning locations $P = (\vec{p}_1, \ldots, \vec{p}_i, \ldots, \vec{p}_n)$ and strategy parameters $S = (\vec{s}_1, \ldots, \vec{s}_i, \ldots, \vec{s}_n)$.
- **Step 2:** Generate $\lambda \geq \mu$ offspring by randomly selecting and recombining ρ individuals from the pool of μ parents:
  - $\vec{p}_i^{\,\text{rec}} = \chi(\vec{p}_1^{\,1}, \ldots, \vec{p}_\rho^{\,1})$ where, $\vec{p}_j^{\,1} \in P$
  - $\vec{p}_i^{\,\text{rec}} = (\chi(p_{i1}, p_{k1}), \ldots, \chi(p_{id}, p_{kd}))$.
- **Step 3:** Mutations:
  - $\vec{p}_i^{\,\text{new}} := \vec{p}_i^{\,\text{rec}} + \vec{z}_i$ where $\vec{z}_i := \sigma(N_0(S_{i1}) \ldots N_0(S_{id}))$
  - $N_\alpha(S)$ returns a Gaussian distributed random value around α with variance S.
- **Step 4:** Evaluate the fitness of all individuals.
- **Step 5:** Select the μ best individuals to serve as parents for the next generation.
- **Step 6:** If the termination criterion is met:
  - STOP,
  - Otherwise, go to Step 2.

□ 2. Basic Particle Swarm Optimization: bPSO

As the basic particle swarm optimizer (bPSO) we use Eberhart's and Kennedy's original PSO version [10]. Inspired by both social behaviour and bird flocking patterns, the particles 'fly' through the solution space and tend to land on better solutions.

The search is performed by a population of particles i, each having a location vector $P = (\vec{p}_{i1}, \ldots, \vec{p}_{ij}, \ldots, \vec{p}_{id})$, and representing a potential solution in a d−dimensional search space. Each particle i also keeps track of its velocity vector, $\vec{v}_i$, which determines the direction and how far a particle will move in the next iteration. The fitness of a particle is determined by an evaluation function $\mathcal{F}(\vec{p}_i)$. Particles move through the search space in discrete time steps. In order to provide a balance between local (with a

higher tendency to converge to solution in close vicinity) and global search (looking for overall good solutions) , an inertia weight ω (step 5) was suggested by Eberhart [4],[5]. In algorithm bPSO, this term is set to a constant value of $\omega = 1$.

bPSO algorithm

- **Step 1:** Initialize the particle population by stochastically assigning locations $P = (\vec{p}_1, \ldots, \vec{p}_i, \ldots, \vec{p}_n)$ and velocities $V = (\vec{v}_1, \ldots, \vec{v}_i, \ldots, \vec{v}_n)$.
- **Step 2:** Evaluate the fitness of all particles: $\mathcal{F}(P) = (\mathcal{F}(\vec{p}_1), \ldots, \mathcal{F}(\vec{p}_i), \ldots, \mathcal{F}(\vec{p}_n))$.
- **Step 3:** Keep track of the locations where each individual had its highest fitness so far:
  - $P = \left(\vec{p}_1^{\,\text{best}}, \ldots, \vec{p}_i^{\,\text{best}}, \ldots, \vec{p}_n^{\,\text{best}}\right)$.
- **Step 4:** Keep track of the position with the global best fitness:
  - $\vec{p}_g^{\,\text{best}} = \max_{\vec{p} \in P} (\mathcal{F}(\vec{p}_i))$.
- **Step 5:** Modify the particle velocities based on the previous best and global best positions:
  - $\vec{v}_i^{\,\text{new}} = \omega * \vec{v}_i + \varphi_1 \left(\vec{p}_i^{\,\text{best}} - \vec{p}_i\right) + \varphi_2 \left(\vec{p}_g^{\,\text{best}} - \vec{p}_i\right)$ for $1 \le i \le n$.
- **Step 6:** Update the particle locations:
  - $\vec{p}_i^{\,\text{new}} = \vec{p}_i + \vec{v}_i^{\,\text{new}}$ for $1 \le i \le n$.
- **Step 7:** If the termination criterion is met:
  - STOP,
  - Otherwise, go to Step 2.

□ 3. Random Particle Swarm Optimization: rPSO

Previous work on PSO suggests that the so−called inertia weight ω should be annealed (dPSO) over time in order to obtain better results [6]. This time−decreasing inertia weight facilitates a global search at the beginning and the later small inertia weight fine tunes the search space. Since the annealed value is dependent on time, the number of iterations must be known in advance. However, in most real−world scenarios, like the soccer kick optimization, it is extremely difficult to get to know the number of necessary iteration steps in advance. The 'random' rPSO version tries to alleviate this problem by assigning a random number to ω (Step 5) in each iteration as follows:

$$\omega = \frac{0.5 + \text{Random}[]}{2}$$

This generates a uniformly distributed random number in the interval $[\frac{1}{4}, \frac{3}{4}]$.

■ 3. Benchmarks

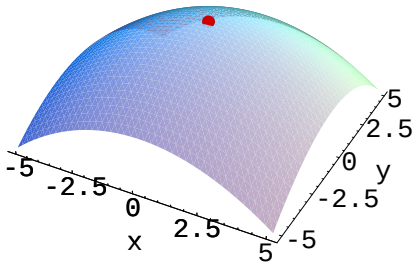
In the light of the No Free Lunch Theorem [16], it is, difficult to identify a clearly superior search or optimization algorithm in any comparison. Therefore our purpose is not to show which one of the three algorithms is outperforming the others in any case, but to find out which of these optimizers is better suited for specific optimization challenges. In particular, we also want to investigate whether an algorithm's performance characteristics in two dimensions−−−where visualization and manual inspection is easiest and most accesible−−−transfers to higher dimensions. We evaluate the performance of (1 + λ) ES scheme and both versions of the particle swarm algorithms on a small set of numerical benchmark functions.

We use the five benchmark functions illustrated and described in more detail below. We explore each of these benchmark search spaces for dimensions $d = 2$, $d = 4$, and $d = 10$. The first three functions are unimodal, that is, with a single global optimum. The last two functions are multimodal, where the number of local maxima increases exponentially with the problem size [15].

f_1 : Sphere

$$f_1 = -\sum_{j=1}^{d} x_j{}^2, \; -5.12 \le x_j \le 5.12, \; \mathcal{F}(\vec{x}^*) = 0.$$

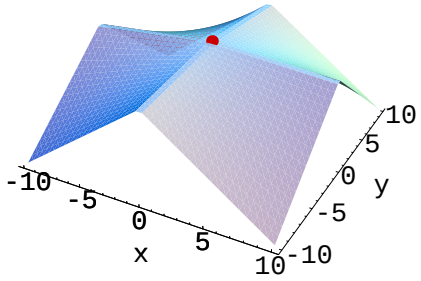
This is a simple, symmetric, smooth, unimodal search space (inverted parabola), and is known to be easily solved by all algorithms. As in our case, it is mainly used to calibrate control parameters of the optimizers.



f_2 : Edge

$$f_2 = -\left(\sum_{j=1}^{d} \left| x_j \right| + \prod_{j=1}^{d} \left| x_j \right|\right), \; -10 \le x_j \le 10, \; \mathcal{F}(\vec{x}^*) = 0.$$

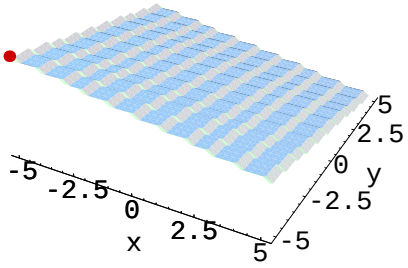
Function f_2 has shared edges making it more difficult to find good solutions around the ridges.



f_3 : Step

$$f_3 = -\left(12 + \sum_{j=1}^{d} \lfloor x_j \rfloor\right), -5.12 \le x_j \le 5.12, \mathcal{F}(\vec{x}^*) = 0.$$

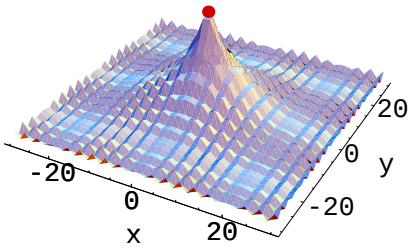
We have included the linear surface function, f_3 ,in order to see whether the alogrithms perform a gradient ascent strategy.



f_4 : Ackley

$$f_4 = -\left(-20\, e^{-0.2\sqrt{\frac{1}{d}\sum_{j=1}^{d} x_j^2}} - e^{\frac{1}{d}\sum_{j=1}^{d} \mathrm{Cos}[2\pi x_j]} + 20 + e\right), -30 \le x_j \le 30, \mathcal{F}(\vec{x}^*) = 0.$$

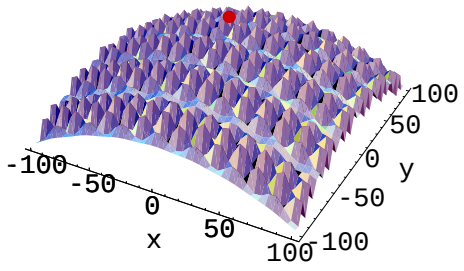
The Ackley function is more difficult as search algorithms tend to settle in any of the local optima, making it more challenging to identify the global optimum.



f_5 : Griewangk

$$f_5 = -\left(1 + \frac{1}{4000}\sum_{j=i}^{d} x_j^2 - \prod_{j=1}^{d} \mathrm{Cos}\left[\frac{x_j}{\sqrt{j}}\right]\right), -100 \le x_j \le 100, \mathcal{F}(\vec{x}^*) = 0.$$

Griewangk's function has hundreds of local optima above the sphere. We included it to compare the algorithms' performances on Griewangk and the sphere function.



■ 4. Experimental Setup

For each of the three optimizers−−−ES, bPSO, and rPSO−−−we performed 20 experiments on each of the five test functions f_1 to f_5 for dimensions d = 2, d = 4, and d = 10. A different random number seed is used for each of the 20 runs. The initial individuals (including those for ES) are uniformly distributed throughout the search space. Each initial population, generated for an ES experiment, is also used for the bPSO and rPSO experiments. This ensures that all comparable runs start with the same initial distribution of individuals. The termination criterion for all runs was to stop when the maximum number of iterations t_{max} = 1,500, was reached. By that time all the optimizers had already reached their convergence phase (Figure 2). The parameter settings for the three algorithms are described in Tables 1, 2, and 3.

population size, n	10
location range, $p_{ij} \in [p_{low}, p_{high}]$	varies
velocity range, $v_{ij} \in [v_{low}, v_{high}]$	10 % of p_{ij}
exploitation rate, φ_1	0.1
exploration rate, φ_1	1

Table 1. bPSO parameter settings

population size, n	10
location range, $p_{ij} \in [p_{low}, p_{high}]$	varies
velocity range, $v_{ij} \in [v_{low}, v_{high}]$	10 % of p_{ij}
exploitation rate, φ_1	1.5
exploration rate, φ_1	1.5

Table 2. rPSO parameter settings

population size (selection pool), $1 + \lambda$	1 + 9
mutation step − size radius	1

Table 3. ES parameter settings

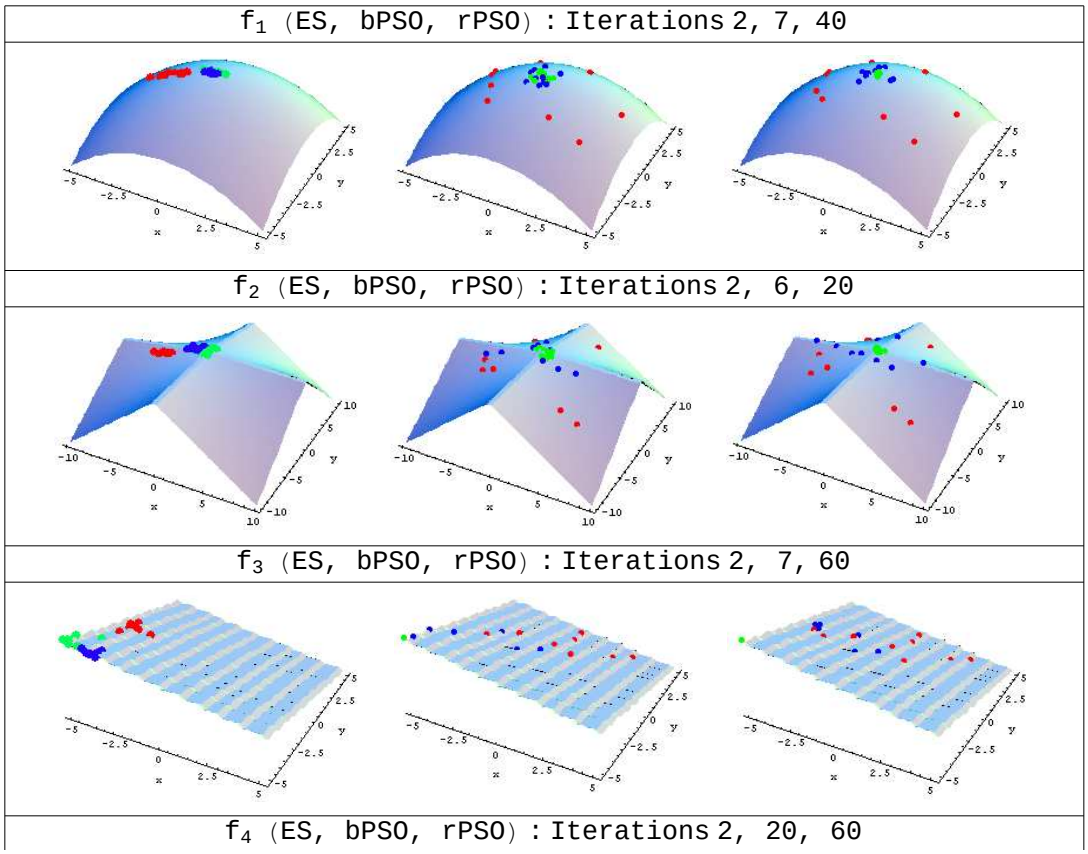
■ 5. Discussion of the Results

The results of the evolution–based optimizer and the two particle swarm optimization techniques are briefly discussed in this section.

□ 1. Phenotype Plots

Figure 1 gives an example of the population dynamics resulting from each of the three algorithms (ES – column 1, bPSO – column 2, and rPSO – column 3) applied over a certain number of iterations. The individuals are represented as dots and in each plot three iterations (red, blue, green) are depicted. In order to achieve a fair comparison, all three algorithms start from the same initial populations. The behaviour of the individuals is seen at different iterations making it easy to compare and contrast the movement of the individuals and study their convergence behaviour. For example, f_1 is plotted at iterations 2 (red), 7 (blue), and 40 (green).

In comparison to the (1+λ) ES scheme, we observe that the particle swarm individuals (both bPSO and rPSO) search the solution space more thoroughly and in multiple directions in comparison to the (1+λ) ES scheme and have higher exploration capabilities. The ES individuals stay close to each other within a certain mutation radius. This is a typical effect of using this particular scheme of ES. The individuals of algorithm ES converge to a local optimum solution for functions f_4 and f_5. This also illustrates the fact that ES individuals exhibit strong local search behaviours, which in this case is mainly due to a relatively small mutation step size.



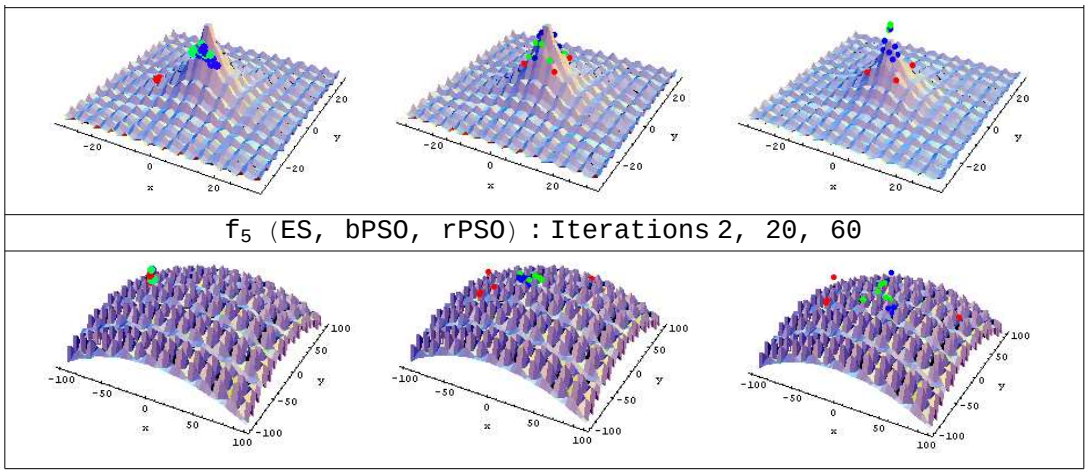


Figure 1. Phenotypical plots for the 2−dimensions versions of the benchmark functions f_1 to f_5 from top to bottom. From left to right the columns show snapshots of typical optimization runs for ES, bPSO, and rPSO. The populations of the first, second, and third snapshot are represented as red, blue, and green spheres respectively.

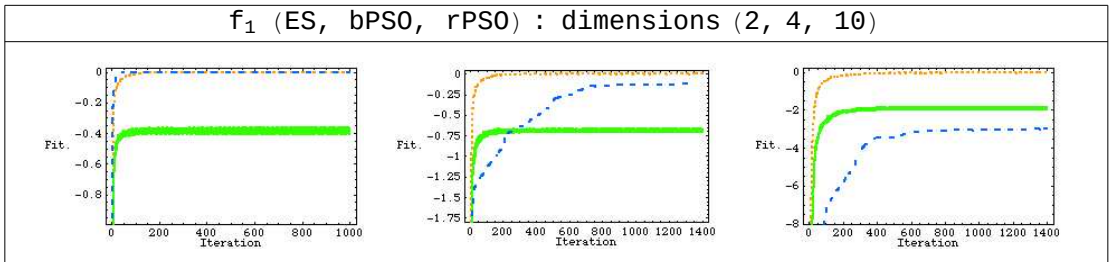
□ 2. Convergence Plots

The convergence plots represent mean fitness values computed over all twenty runs for all 1,500 iterations. This graph helps in demonstrating the convergent behaviour of the individuals of a particular algorithm and also illustrates which of the three algorithms has the fastest fitness convergence speed. Figure 2 summarizes the results are shown for d = 2 (column 1), d = 4 (column 2), and d = 10 (column 3).

In almost all cases, algorithm ES is comparable to rPSO. Algorithm bPSO turns out to be the slowest in terms of convergence speed for d = 2. However, as the number of dimensions increases (d = 4), the convergence rate of ES decreases, and in ten dimensions, ES converges more slowlyand towards a lower fitness values.

All the three algorithms show relatively steep ascents during the first 100 or 200 iterations. After which rPSO seemed to rapidly level off without making any further progress (for most of the test cases) during the rest of the simulation i.e., the swarm stagnates (no changes observed in terms of finding a better fitness value). For instance, on f_1, particle swarms stagnate and flatten out without any further improvements. However, the convergence rate of ES on function f_3, gradually slowed down but did not completely level off, which indicates that if it were allowed to run longer it may have discovered better solutions.

Another observation made for function f_5 in two dimensions is that rPSO has the slowest convergence rate. This is in line with the results of the phenotype plot (Figure 1) where the particles do not converge to one location.



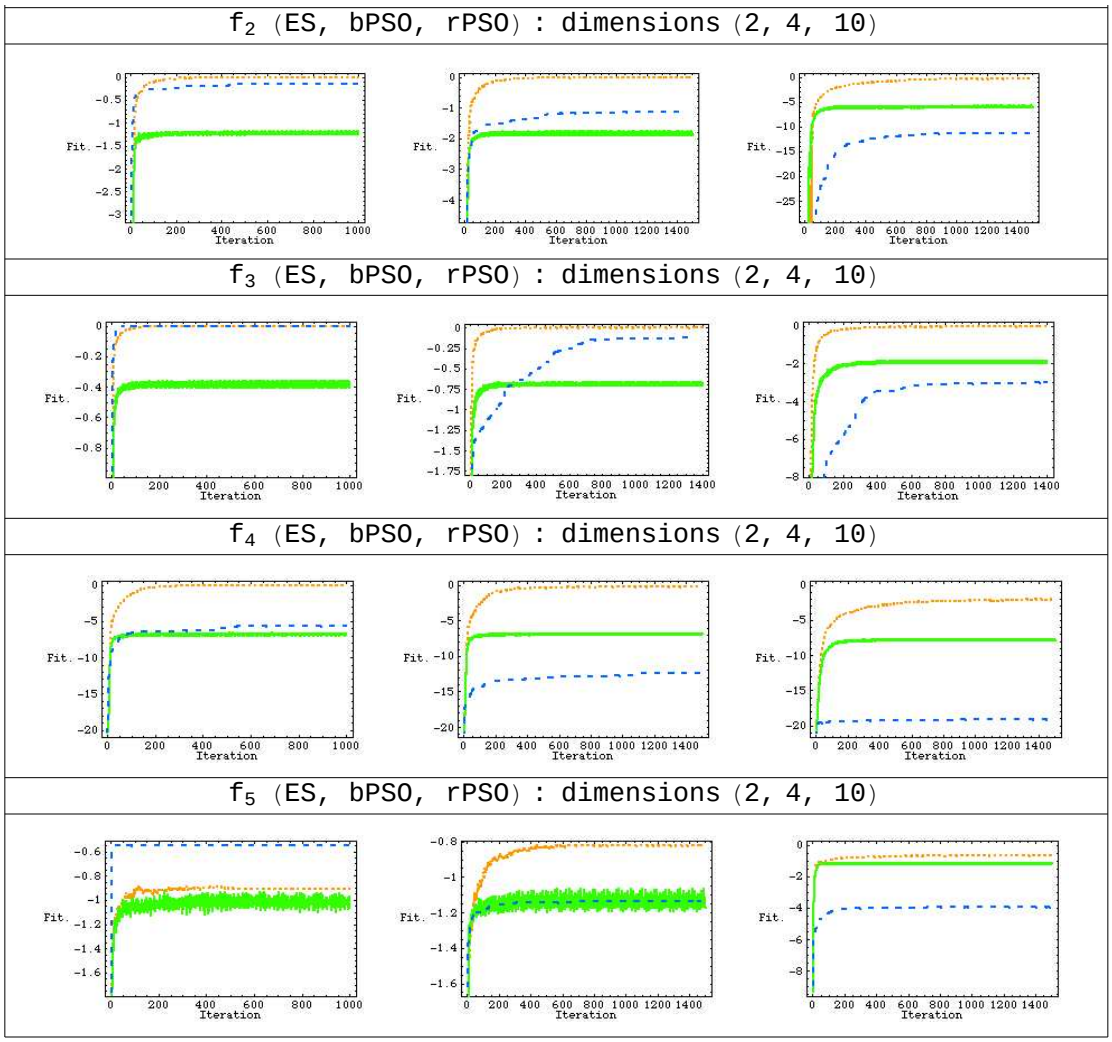


Figure 2. Fitness plots of the benchmark functions f_1 to f_5 from top to bottom. From left to right the columns illustrate the convergence behaviours of the three algorithms (ES: blue, rPSO: orange, bPSO: green) in 2, 4, and 10 dimensions.

□ 3. Success Plots

The best fitness value obtained at the end of each run is illustrated in Figure 3. Here, for each function, the fitness values of all 20 runs are plotted in ascending order from left to right. Therefore, each graph displays the success rate of each algorithm on a particular function. The best (right−most point), worst (left−most point), and mean fitnesses can also be easily derived from these graphs.

In all 20 runs, algorithm ES finds worse solutions than both PSO algorithms for $d = 2$, 4, and 10. This is observed by the left−most blue point in Figure 3. This is in line with the results of the phenotype plots (Figure 1), especially for the multimodal functions f_4 and f_5, where the ES individuals are unsuccessful in finding the global optimum. The higher exploration capabilities of the PSO algorithms seem to facilitate the discovery of better solutions in comparison to the local ES scheme.

In the phenotype plots (Figure 1) one can also observe that the bPSO particles do not converge to one solution only. However, there is always at least one particle that finds the global optimum. Therefore, both PSO algorithms (bPSO and rPSO) are comparable

in terms of finding the best solution, as illustrated by the right most green and orange points in Figure 3.

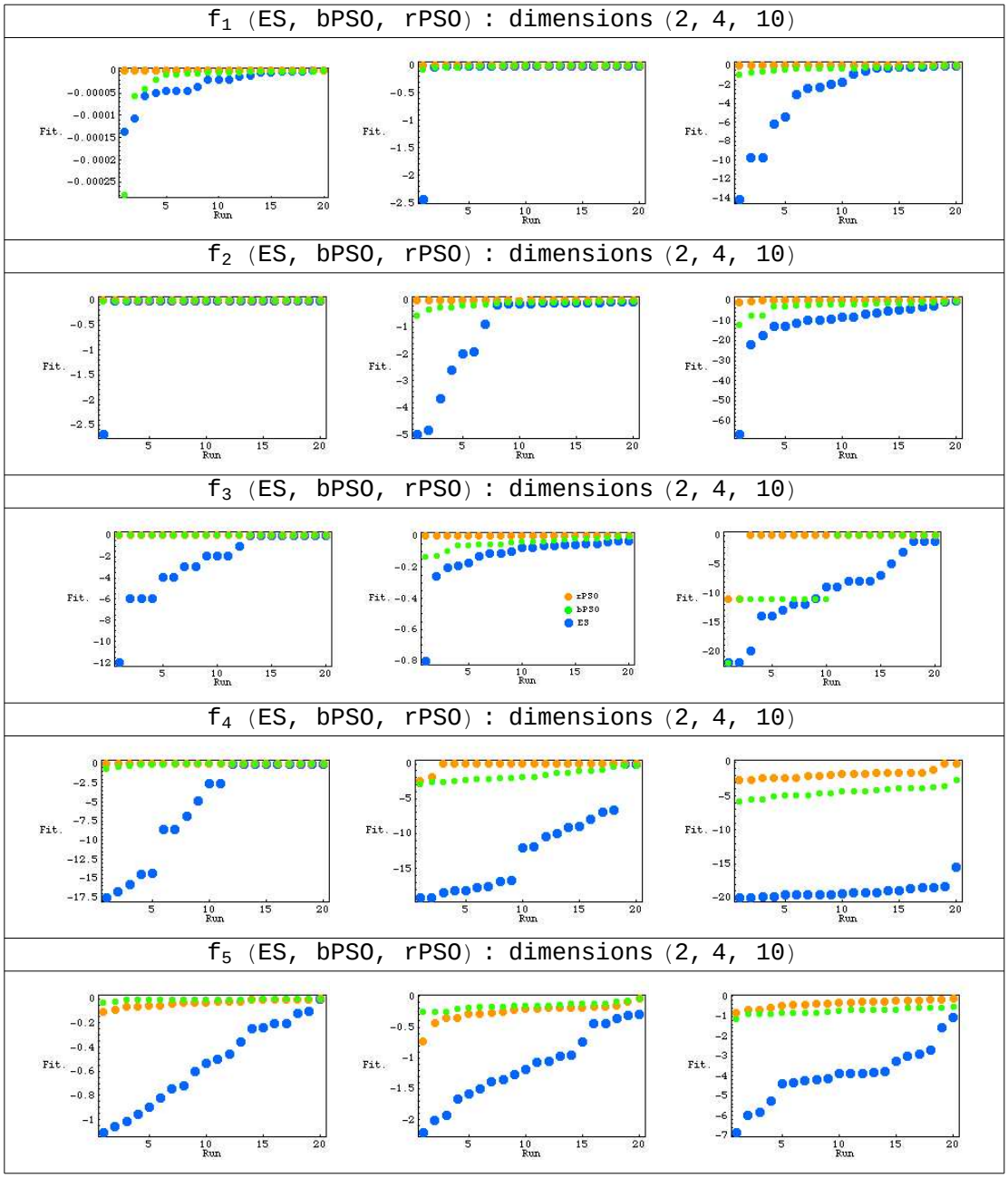


Figure 3. Success ratio plots of the benchmark functions f_1 to f_5 from top to bottom. From left to right the columns show the best fitness value obtained at the end of each run of the three algorithms (ES: blue, rPSO: orange, bPSO: green) in 2, 4, and 10 dimensions.

4. Parameter Range Plots

For the following analyses we only look at algorithm performance on the 10−dimensional benchmarks. In Figure 4 we visualize how for each variable range changes during the course of the evolutionary search. For example, in the first row of Figure 4 the vertical bars represent the range for each of the ten variables, over all iterations, limited to all those solutions that have a fitness of at least $\tau \geq -5$ (the highest fitness is zero). Knowing how the value ranges change is important especially when exploring real−world optimization problems, since it can provide insights on whether the fitness function is sensitive with respect to a particular variable or not. It also clearly visualizes the stage of convergence in each dimension.

Figure 4 shows that algorithm ES maintains a high parameter range in comparison to the particle swarm algorithms. This is in line with the results of Figure 2, where ES has the slowest convergence speed for $d = 10$. Evolution strategies seem to consistently keep wider parameter ranges. Both particle swarm algorithms show comparable ranges.

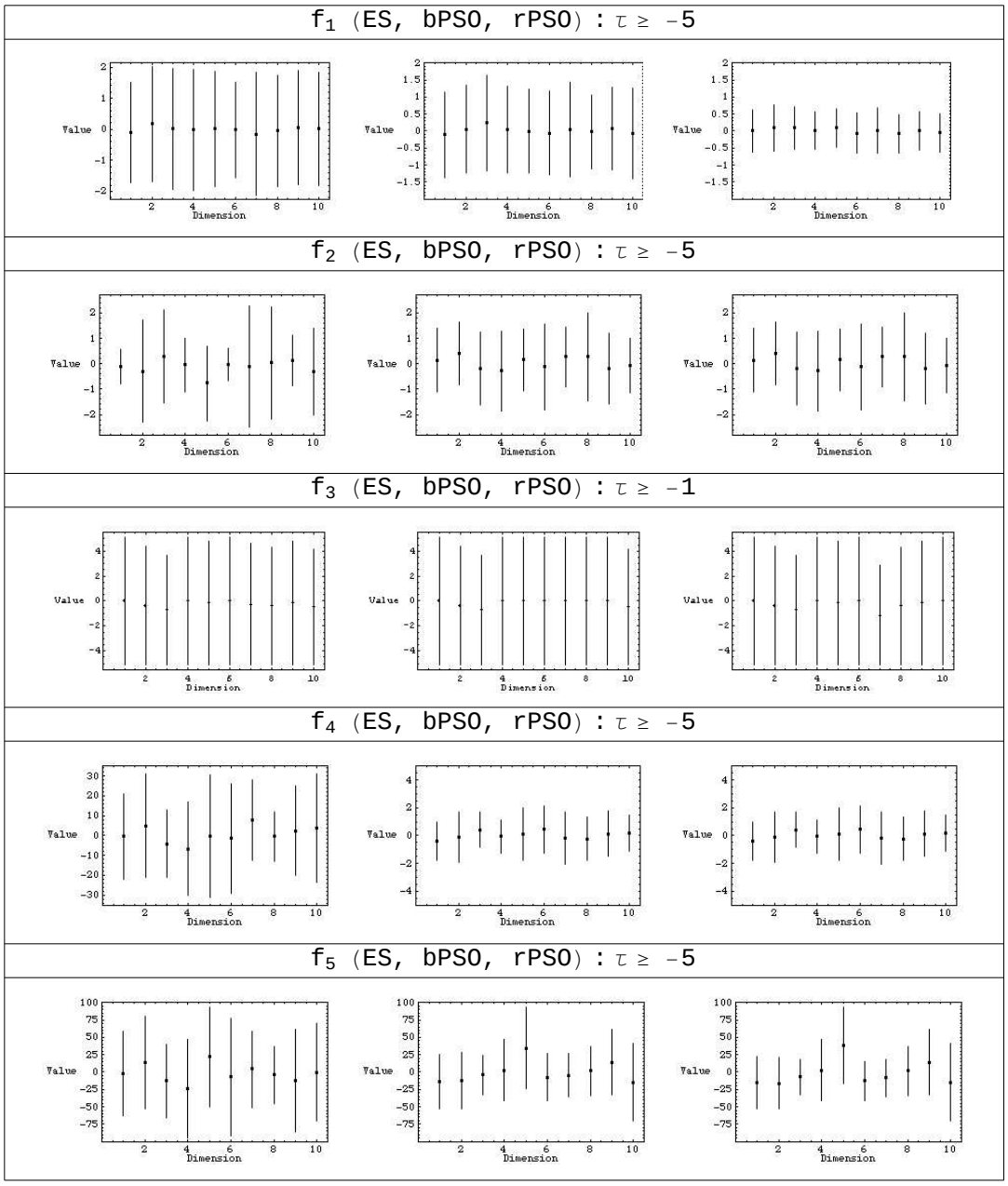


Figure 4. Parameter range plots for the 10−dimensional versions of the benchmark functions f_1 to f_5 from top to bottom. From left to right the columns show the parameter range for each of the ten dimensions over a certain threshold value (τ) for ES, bPSO, and rPSO.

■ 6. "Evolutionary Swarms" web*Mathematica* website

The comparisons conducted provide knowledge and insights about the algorithms, while illustrating the movement of the individuals in the solution space (Figure 1). They point out various characteristics like the convergence speed and the success an algorithm has in finding good solutions. The strengths and weaknesses of the algorithms are exploited, for example, PSO algorithms have a higher exploration rate whereas the $(1 + \lambda)$ scheme of ES depicts a local search scheme.

As stated earlier, we have created a set of notebooks to compare $(1 + \lambda)$ scheme of ES and both the particle swarm optimizers on the benchmark functions ($f_1 - f_5$). We provide the interface to the users for creating the above graphs (Figure 1, Figure 2, Figure 3, and Figure 4). We have converted these notebooks to a web*Mathematica* interface. This interactive site provides the hands−on experience and an experimental environment through a selection of ten benchmark functions along with the visualization tools (for generating the images).

This site currently consists of particle swarms (PSO) and its three variants—basic particle swarms (bPSO), random particle swarms (rPSO), and particle swarms with decreasing inertia weight (dPSO). We also implemented both the simple schemes of ES, the $(1 + \lambda)$ scheme of ES and the $(1 , \lambda)$ scheme of ES. The generalized evolution strategies, $(\mu + \lambda)$ and (μ , λ), as described in Section 2.

As it is in general difficult to know the settings of various parameters, we provide suggestions for different settings. This will help the users to gain further knowledge regarding these optimizers.

The website can be accessed at:
http://oak.cpsc.ucalgary.ca:8080/webMathematica/optimization/optimization.jsp

■ 7. Conclusion

The best way of understanding evolutionary− and swarm−based algorithm heuristics is through practical experience, in particular on smaller scale problems. The 'Evolutionary Swarms' site that we have developed will be integrated with the collection of notebooks from the EVOLVICA package. This database of notebooks, along with the swarm algorithms, provide a large experimental and inquiry platform for introducing evolutionary and swarm−based optimization techniques to those who wish to further their knowledge in the evolutionary computation domain. Making these notebooks available through a web*Mathematica* site means that anyone with access to the newly built web pages will have instant access to a wide range of optimization algorithms.

■ References

[1] Cole G., Gerristen K., Influence of mass distribution in the shoe and plate stiffness on ball velocity during a soccer kick, *Adidas−Salomon AG*, 2002.

[2] Cole G., Gerristen K., Optimal mass distribution and plate stiffness of football shoes, *Adidas−Salomon AG*, 2002.

[3] Cole G., Gerristen K., Influence of medio−lateral mass distribution in a soccer shoe on the deflection of the ankle and subtabular joints during off−centre kicks, *Adidas−Salomon AG*, 2003.

[4] Eberhart R.C. andShi Y., Parameter selection in particle swarm optimization. *Evolutionary programming VII: Proceedings of the seventh annual conference on evolutionary programming*, 1998.

[5] Eberhart R.C. andShi Y., A modified particle swarm optimizer. *Proceedings of IEEE congress on evolutionary computation (CEC 1998)*, 1998.

[6] Eberhart R.C. and Shi Y., Comparion between genetic algorithms and particle swarm optimization. *Evolutionary programming VII: Proceedings of the seventh annual conference on evolutionary programming,* 1998.

[7] Goldberg D., *Genetic Algorithms in search, optimization, and machine learning,.* Addison−Wesley 1989.

[8] Jacob C., Khemka N., Particle Swarm Optimization in Mathematica An Exploration Kit for Evolutionary Optimization, *IMS'04, Proc. Sixth International Mathematica Symposium*, Banff, Canada (2004).

[9] Jacob C., *Illustrating Evolutionary Computation with Mathematica,* Morgan Kaufmann Publishers, 2001.

[10] Kennedy J. and Eberhart R.C., Particle swarm optimization, *Proceedings of IEEE International Conference on Neural Networks*, 1995.

[11] Khemka N., *Comparing PSO And ES: benchmarks and application*, M.Sc. Thesis, UofCalgary, December 2005.

[12] Khemka N., Jacob C., Cole.G., Making soccer kicks better: A study in PSO and ES, *Proceedings of IEEE congress on evolutionary computation (CEC 2005),* 2005.

[13] Koza J., *Genetic Programming − On the programming of computers by means of natural selection*, MIT press, 1992.

[14] Rechenberg I., Evolution Strategies: Nature's way of optimization, Optimization methods and applications, possibilities and limitations, 47, *Lecture Notes in Computer Science*,1989.

[15] Schwfel H.P., *Evolution and optimum seeking,* John Wiley and Sons, 1995.

[16] Wolpert D.H. and Macready W.G., No free lunch theorems for search, Santa Fe Institute, 1995.