CHASING THE SWARM: A PREDATOR-PREY APPROACH TO FUNCTION OPTIMISATION

Arlindo Silva [1,2] Ana Neves [1,2] Ernesto Costa [2]

[1] Escola Superior de Tecnologia
Instituto Politécnico de Castelo Branco
Av. do Empresário, 6000 Castelo Branco, Portugal
Phone: 351 272 325 653
Email: {arlindo, dorian}@est.ipcb.pt

[2] Centro de Informática e Sistemas da
Universidade de Coimbra
Pólo II, Pinhal de Marrocos, 3030 Coimbra, Portugal
Phone: 351 239 790 000
Email: ernesto@dei.uc.pt

***Abstract**: In this paper we describe our approach to predator-prey optimisation, a form of particle swarm optimisation where new particles called predators are introduced. The objective of predator-prey optimisation is to use predator particles to help avoiding premature convergence to sub-optimal solutions in particle swarm optimisers. The swarm particles (prey particles) are repelled by predators, which in turn are attracted to the best individuals in the swarm. The resulting interactions make total convergence difficult to the swarm, maintaining diversity in the population. First results of this new approach on several benchmark functions are presented and the performance of the algorithm is compared to the performance of the standard particle swarm optimiser.*

Keywords: Particle swarm optimisation, Predator-prey optimisation

1 - Introduction

The particle swarm algorithm has been originally presented by [Kennedy95] as a population based function optimiser in the *n*-dimensional space of real numbers. Bird flock flight simulations initially inspired the algorithm and biological inspiration is still present in the current denomination. The swarm or flock metaphor applies to the particle swarm optimiser (PSO) in the way particles fly in a somewhat coordinated way through a *n*-dimensional space (the space of parameters of the function being optimised) in search of some desired place (function optimum).

While being considered a form of evolutionary algorithm, there is no use of genetic operators like mutation or recombination, as is the case in other evolutionary paradigms such as evolutionary programming, evolutionary strategies or genetic algorithms. Explicit selection is also not present. Instead, in each iteration, the position of every particle in the search space is updated accordingly to the particle's velocity. The velocity of a particle in a given iteration is a function of the velocity in the previous iteration, its best previous position in the search space and the best previous position in the search place of all of the particle's neighbours. The behaviour of a particle in the swarm is the result of balancing the desire of flying towards the best point in the search place according to its own experience or conforming to the swarm knowledge of where the current best point is. For an extensive discussion of the cultural model behind the PSO, as well as of the PSO itself and several variants see [Kennedy01].

While the PSO has revealed itself capable of competing with other evolutionary techniques, namely the standard genetic algorithm (GA), it has been noted [Angeline98] that the original PSO had difficulties controlling the balance between exploration (global investigation of the search place) and exploitation (the fine search around a local optimum). According to [Angeline98], the PSO, while quickly converging towards an optimum in the first iterations, has problems when it comes to reach a near optimal solution. To solve this [Shi98] introduced the use of a linear decreasing inertia weight, reminiscent from the temperature parameter in simulated annealing. In [Shi99] results of using the PSO with the inertia weight on several benchmark functions are presented and is concluded that, while performing significantly better that the original PSO, lacks global search ability at the end of the run. Other approaches to this problem include the proposal of hybrid PSO-GA models [Lovbjerg01], which presented some improvements in performance but only in one of the test functions.

We present here a new approach to balancing exploration and exploitation in PSO, by introducing a second population of particles, which we call *predators*. Predators have a different dynamic behaviour from the swarm particles (which we call *prey*). They are attracted to the best individuals in the swarm, while the other particles are repelled by their presence. Controlling the strength and frequency of the interactions between predators and prey, we can influence the balance between exploration and exploitation and maintain some diversity in the population, even when it is approaching convergence, thus reducing the risk of convergence to local sub-optima. We also present and discuss the first experimental results obtained by comparing this model with standard PSO in a set of benchmark functions.

The remainder of is article is organized in the following way: In the next section we briefly describe the standard PSO model. Section 3 is devoted to the description of the predator-prey optimiser (PPO). Description of the experimental setup is made in section 4, while the experimental results are presented and discussed in section 5. Finally, section 6 is dedicated to the presentation of some conclusions and objectives for future work.

2 -The Particle Swarm Optimiser

In particle swarm optimisation a population of point particles "fly" in an *n*-dimensional real number search space, where each dimension corresponds to a parameter in a function being optimised. The position of the particle in the search space is represented by a vector *X*. The velocity of the particle, i.e., its change in position, is represented by a vector *V*. The particle "flies" in the search space by adding the velocity vector to its position vector in order to change its position.

V determines the particle's trajectory and depends on two "urges" for each particle *i*: flying towards its best previous position and flying towards its neighbours' best previous position. Different neighbourhood definitions have been tried [Kennedy99]; here we assume that every particle is a neighbour to every other particle in the swarm. The equations for updating the position and velocity for some particle *i* are the following:

$$\begin{cases} V_i(t) = wV_i(t-1) + \varphi_{1i}(P_i - X_i(t-1)) + \varphi_{2i}(P_g - X_i(t-1)) \\ X_i(t) = X_i(t-1) + V_i(t) \end{cases} \tag{1}$$

In the above formula φ_1 and φ_2 are random numbers distributed between 0 and an upper limit and different for each dimension in each individual, P_i is the best position particle *i* has found in the search space and *g* is the index of the best individual in the neighbourhood. The velocity is usually limited in absolute value to a predefined maximum, V_{max}. The parameter *w* is the linear decreasing weight, which, during the run, decreases from $w_{\max}$ to $w_{\min}$. The swarm is usually run for a limit number of iterations or until an error criterion is met.

3 -The Predator-Prey Optimiser

Our motivation for developing the predator-prey model was mainly to introduce a mechanism for creating diversity in the swarm at any moment during the run of the algorithm, not depending on the level of convergence already achieved. This would allow the "escape" of particles even when convergence of the swarm around a local sub-optimum had already occurred. A second, and less practical, motive was to maintain the biological metaphor. Other mechanisms could perhaps have been used to the same effect, but it seemed more appropriate to introduce a mechanism that also has some parallel in nature. The predator-prey model is inspired in the hunt in nature of animals grouped in flocks by one or more predators. When chased, animals have more difficulty to stay around their most preferable places (better pastures, water sources…) and have to search for other locations, free of predators and perhaps even better. This is the effect we want to model in our algorithm, where the metaphorical better pastures are the functions' local sub-optima.

In the present state of development of the predator-prey optimiser, only one predator is used. The predator's objective is to pursue the best individual in the swarm, i.e. the individual that has found the best point in the search space corresponding to the function being optimised. The predator update equations are:

$$\begin{cases} V_p(t) = \varphi_4(X_g(t-1) - X_p(t-1)) \\ X_p(t) = X_p(t-1) + V_p(t) \end{cases} \tag{2}$$

φ_4 is another random number distributed between 0 and an upper limit and X_g is the present position of the best particle in the swarm. The upper limit on φ_4 allows us to control how fast the predator "catches" the best individual.

The influence of the predator on any individual in the swarm is controlled by a "fear" probability P_f, which is the probability of a particle changing its velocity in one of the available dimensions due to the presence of the predator. The dimension where the change will occur is randomly chosen. For some particle *i*, if there is no change in the velocity in a dimension *j* the update rules in that dimension still are:

$$\begin{cases} v_{ij}(t) = wv_{ij}(t-1) + \varphi_{1ij}(p_{ij} - x_{ij}(t-1)) + \varphi_{2ij}(p_{gj} - x_{ij}(t-1)) \\ x_{ij}(t) = x_{ij}(t-1) + v_{ij}(t) \end{cases} \tag{3}$$

But if the predator influences the velocity in dimension j, the rule becomes:

$$\begin{cases} v_{ij}(t) = wv_{ij}(t-1) + \varphi_{1ij}(p_{ij} - x_{ij}(t-1)) + \varphi_{2ij}(p_{gj} - x_{ij}(t-1)) + \varphi_{3ij}D(d) \\ x_{ij}(t) = x_{ij}(t-1) + v_{ij}(t) \end{cases} \tag{4}$$

The fourth term in the first equation in (4) quantifies the repulsive influence of the predator by modifying the velocity adding a value that is a function of the difference between the position of the predator and the particle. *d* is the averaged sum of the absolute values of the distances in each dimension. *D(x)* is an exponential decreasing distance function defined as:

$$D(x) = ae^{-bx}$$

D(x) makes the influence of the predator grow exponentially with proximity. The objective of its use is to introduce more perturbation in the swarm when the particles are nearer the predator, which usually happens when convergence occurs. When the distance is bigger (e.g. during the initial exploration phase of the swarm, when *w* is still big), he predator's influence is smaller and usual swarm dynamics take control. The *a* and *b* parameter define the form of the *D* function: *a* represents the maximum amplitude of the predator effect over a prey and *b* permits to control the distance at which the effect is still significant.

The predator effect was designed to take advantage of the use of *w* as an inertia parameter in the swarm update equations. The idea is to lower the values of *w*, thus forcing a faster convergence, while relying on the predator to maintain population diversity.

4 -The Experimental Setting

To facilitate comparison and analysis of the results we tested the performance of the predator-prey model in six non-linear functions, the first four frequently used as benchmarks in swarm particle literature [Angeline98, Shi99, Lovbjerg01, Kennedy01], while f_5 and f_6 are commonly used as benchmarks for evolutionary algorithms (e.g. [Muhlenbein93]).

$$f_1(x) = \sum_{i=1}^{n} x_i^2$$

$$f_2(x) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$$

$$f_3(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$$

$$f_4(x) = \frac{1}{4000}\sum_{i=1}^{n} (x_i - 100)^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i - 100}{\sqrt{i}}\right) + 1$$

$$f_5(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$$

$$f_6(x) = 418.9829n + \sum_{i=1}^{n} - x_i \sin\left(\sqrt{|x_i|}\right)$$

In the functions above, *x* is a real number, *n*-dimensional vector and x_i is the *i*th element of that vector. f_1 is the generalized sphere function, a simple unimodal function; f_2 is the generalized Rosenbrock function, a unimodal function known to be hard to optimise; f_3 is the generalized Rastrigin function, f_4 is the generalized Griewank function, f_5 the generalized Ackley function, three multimodal functions with many local minima set around the global minima in an unimodal macrostructure; and, finally, f_6 which is a multimodal function, designed to be difficult for evolutionary algorithms, with the local minima set far away from each other and the global minimum located at the boundary of the search space. All these functions are used as minimization problems.

Two sets of experiences were run with the standard particle swarm optimiser. In one set the linear decreasing weight limits were set to w_{min}=0.4 and w_{max}=0.9, which are standard values recommended in PSO literature [Shi98]. In the second set, the same limits were set to w_{min}=0.0 and w_{max}=0.7. to allow comparison of results with the predator-prey optimiser results, which also used these values. This weight limits were chosen to implement the objective we mentioned before of using the weight limits to force a faster convergence, with the predator-prey effect being expected to provide the lacking population diversity and avoid convergence to sub-optima. In all experiments φ_1 and φ_2 limits were set to 2.0 and V_{max} as set to the same value as X_{max}. In PPO φ_3 was set to 1.0 and φ_4 to 0.1. The *a, b,* and P_f values were, for each function, empirically and after a small set preliminary experiments set to the values presented in Table 1. No systematic effort was made to determine optimum values for these parameters. Larger values of *a* were used for the unimodal functions, in the hope that a lucky trajectory could lead the search to near the global optimum, and for the f_6 function since its minima are far apart. *b* was equally set for all functions to a value of $10.0/X_{max}$ which seemed promising after the preliminary experiments. Lower values of P_f were used with the unimodal functions, were it seemed that diversity was not as needed as in multimodal ones.

Table 1: Parameter values for function used in the predator-prey experiments.

Function	a	b	P_f
f_1	$2.0X_{max}$	$10.0/X_{max}$	0.001
f_2	$2.0X_{max}$	$10.0/X_{max}$	0.001
f_3	$0.1X_{max}$	$10.0/X_{max}$	0.04
f_4	$0.1X_{max}$	$10.0/X_{max}$	0.06
f_5	$0.1X_{max}$	$10.0/X_{max}$	0.04
f_6	$2.0X_{max}$	$10.0/X_{max}$	0.06

Table 2: Search space and initialization limits for each test function.

Function	Search Space	Init. Limits
f_1	$[-100,100]^n$	$[50,100]^n$
f_2	$[-100,100]^n$	$[15,30]^n$
f_3	$[-10,10]^n$	$[2.56,5.12]^n$
f_4	$[-600,600]^n$	$[300,600]^n$
f_5	$[-500,500]^n$	$[-500,500]^n$
f_6	$[-30,30]^n$	$[10,20]^n$

For facilitating comparisons with previous works the algorithm was run with three different dimensions for each function: 10, 20 and 30. An iteration limit was set at 1000 for 10 dimensions, 1500 for 20 dimensions and 2000 for 30 dimensions. A size 20 population was used in all experiments. Care was taken to initialise the population in an asymmetric way for all functions except for f_6 , which, since its structure is not symmetrical around a local minimum situated near the origin, does not benefit from a symmetric initialisation. Table 2 presents the search space and initialisation limits for each of the test functions. For each experimental setting 200 runs of the algorithm were performed.

5 – Experimental Results

Table 3 resumes the results obtained in the sets of experiments described in the previous section. For each experimental setting we present a 90% confidence interval for the average the best solution found in the limit number of iterations, over the 200 runs. The PSO1 column shows the results for the particle swarm optimiser with w decreasing from 0.9 to 0.4, the PSO2 shows the results for the same optimiser with w decreasing from 0.5 to 0.0 and the PPO column shows the results for the predator-prey optimiser. Figure 1 presents graphs illustrating the evolution of average fitness over the run for the six benchmark functions. While graphs are presented only for the dimension 30 experiments they are still representative of the overall behaviour of the optimisers.

Table 3: Confidence intervals at 90% for the average best solutions of each experimental setting over 200 runs.

Func.	Dim.	Iter.	PSO1	PSO2	PPO
f_1	10	1000	1,15E-20±5,83671E-21	5,21E-73±1,02090E-73	1,43E-34±1,22751E-34
	20	1500	5,00E-12±2,28482E-12	2,12E-41±3,13233E-42	9,90E-26±1,03891E-26
	30	2000	4,13E-08±1,30741E-08	6,22E-25±1,01824E-25	5,32E-21±3,76824E-21
f_2	10	1000	118,83096±34,02262	51,14766±16,44399	39,73008±13,55563
	20	1500	185,02141±43,49773	68,89530±17,55632	67,46585±14,97499
	30	2000	241,34265±52,65796	155,13395±35,23946	163,97916±37,66825
f_3	10	1000	5,27192±0,37085	8,22332±0,60453	0,23928±0,07607
	20	1500	23,24823±1,00934	36,63928±1,78038	3,08763±0,38998
	30	2000	48,58373±1,72605	81,17846±3,01176	10,74409±0,93099
f_4	10	1000	0,09822±0,00668	0,08766±0,00636	0,06428±0,00434
	20	1500	0,02836±0,00387	0,02868±0,00338	0,02189±0,00295
	30	2000	0,01585±0,00252	0,02794±0,01233	0,01334±0,00258
f_5	10	1000	2,564E-11±6,03359E-12	0,06941±0,04307	7,832E-08±1,23948E-08
	20	1500	0,00823±0,01613	0,47375±0,09981	1,84E-06±2,68697E-07
	30	2000	0,21048±0,07028	1,08448±0,14345	1,252E-05±1,71359E-06
f_6	10	1000	411,13915±23,70806	689,12034±31,68839	93,73165±13,81862
	20	1500	1202,35393±47,89437	1991,20137±62,02794	380,48637±25,46739
	30	2000	2305,19333±68,72059	3426,18449±77,36661	724,70529±37,99708

A first observation that can be made from table 3 concerns the difference in performance between the PSO1 and the PSO2 optimisers. The only difference between them is that the PSO2 has a "shorter" cooling schedule, with the linear decreasing weight varying from 0.5 to 0.0 while in the PSO1 it varies from 9.0 to 0.4 in the same number of iterations. This should result in an increased urge to converge in the second optimiser. In terms of performance, what can be seen is that the PSO2 performs poorer than the PSO1 in the generality of the experiences involving multimodal functions. In figure 1 we can see that the PSO2 converges faster than the PSO1, but gets more frequently caught in local minima. Since in the first two functions these local minima don't exist, the PSO1 easily outperforms the PSO2.

Since the PPO uses the same cooling schedule as the PSO2, if there is a difference in performance, it must be the result of the predator-prey mechanism. Both from table 1 and figure 1 can be concluded that indeed there is a difference in performance and that that difference is significantly favourable to the predator-prey optimiser.

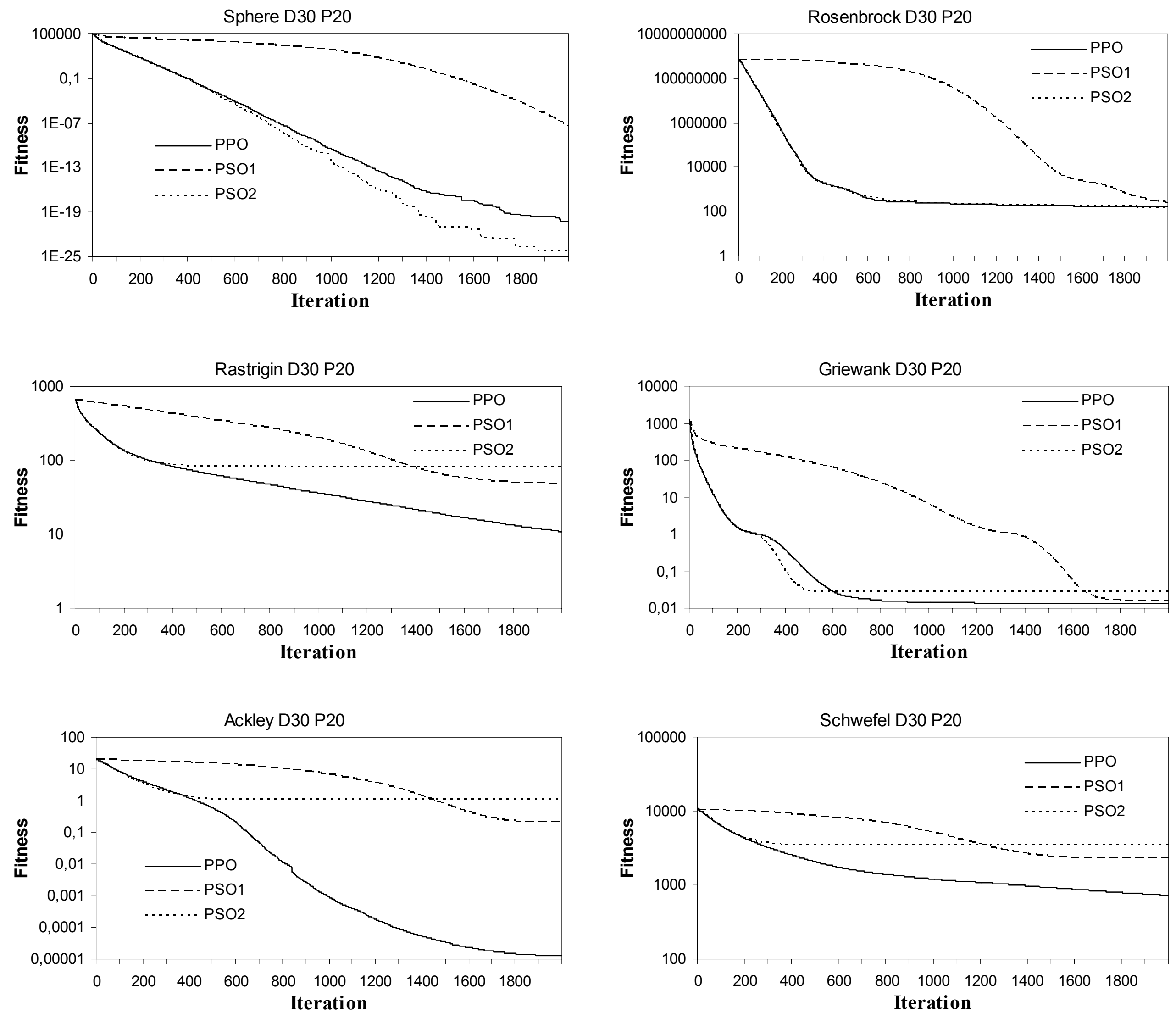

Figure 1: Graphs illustrating evolution of average fitness of PPO versus PSO approaches for the six test functions with dimension 30. Fitness is presented in logarithmic scale.

In the generalized sphere function all optimizers find the optimum very fast, with the PSO2, as expected, being the one to converge faster. The increased diversity of the PPO here only serves to delay its convergence, which still is almost as fast as that of the PSO2. In the Rosenbrock function, being also a unimodal function, the PSO2 is again the first to converge, followed closely by the PPO. As expected, the PPO does not have a better performance than a swarm particle optimizer with a "short" cooling schedule in the two unimodal functions. This is easily explained by the very purpose of PPO: increasing diversity, while it should constitute a useful mechanism against being caught in local minima, does not facilitate convergence in a landscape with only one optimum.

The Rastrigin function was the first multimodal function in which we tested the PSO and it can be seen that the predator-prey approach clearly performs better that both the PPO approaches in the three experimental settings. The PSO not only found, on average, fitter individuals at the end of the run, but was also the first to converge, as can be seen in figure 1. From figure 1 can also be concluded that, giving the fitness curve of the PPO, the optimiser was likely to find even better solutions, if it had more iterations to go, while both PSO approaches seem to have stagnated at the end of the run. In the Griewank function the PPO didn't find significantly better solutions than the PSO1, but is a lot quicker to converge. While the PSO2 converges even quicker, the solutions found are on average worse than the ones found by both the PSO1 and the PPO2. In the Ackley function, the predator-prey optimiser is again the first to converge and, while at 10 dimensions the solutions found are not significantly better that the ones found by the PSO1, at 20 and 30 dimensions the difference is already significant.

Those three functions share a common macro-structure, a global optimum surrounded by a multitude of local sub-optima arranged in a unimodal way towards the optimum, which clearly benefits the characteristics of the PPO over the PSO. The changes in velocity introduced by the predator-prey effect are indeed used, as expected, to "jump" from one minimum to the other, thus allowing the PPO to find better solutions than the standard PSO. And even if better solutions are not found, as it happens in the Griewank function, our previous assertion that the predator-prey mechanism would allow an increase in convergence pressure, materialized by the changes in the inertia weight limits, is validated by the faster convergence of the PPO in the three previous functions.

The last function has a different macro-structure than the previous three. The sub-optima do not surround the global optimum, but are located far in the search space. The optima itself is not situated at, or near, the origin of the referential, but at the boundary of the search space. This structure easily gets the swarm based approaches trapped in the local minima and it can be seen both from table 3 an figure 1 that the three optimisers perform rather poorly in the optimization of this function. But, from the three, the PPO is again the one with the better performance, finding significantly better solutions than both the PSO approaches.

6 – Conclusions and Future Work

We presented a new swarm particle based function optimizer, inspired in the natural relation between predator and prey. A predator-prey mechanism was implemented in addition to the standard swarm particle optimiser, obtaining a new algorithm that was tested with six benchmark functions against the standard PSO. The experimental results allow us to conclude that, while the PPO does not have a better performance than the PSO when it comes to the optimization of unimodal functions, the opposite happens when multi-modal ones are being optimized. In the experiments carried out the PPO performed significantly better than the standard PSO in the optimization of four benchmark multimodal functions, either by finding, on average, better solutions an the end of the runs, or by finding solutions of similar quality but with a higher convergence rate.

In the experimental work presented here, the parameters of the PPO were empirically defined. One of the next steps in our work will be a systematic study of these parameters' influence in the algorithm's performance, so that a set of heuristics can be defined to help in the choice of the most correct parameters for the optimisation of a given function. We are also aiming to develop a multi-predator sub-population version of the predator-prey optimiser.

Acknowledgements

This work was partially financed by the Portuguese Ministry of Science and Technology under the Program POSI.

Bibliography

[Kennedy95] Kennedy, J. and Eberhart, R. C., "Particle swarm optimisation", Proc. IEEE International Conference on Neural Networks. Piscataway, NJ, pp. 1942-1948, 1995.

[Kennedy01] Kennedy, J., Eberhart, R. C., and Shi, Y., "Swarm intelligence", Morgan Kaufmann Publishers, San Francisco. 2001

[Angeline98] Angeline, P. J., "Evolutionary optimization versus particle swarm optimization: philosophy and performance differences", The Seventh Annual Conf. on Evolutionary Programming. 1998.

[Shi98] Shi, Y. and Eberhart, R. C., "Parameter selection in particle swarm optimisation" Evolutionary Programming VII: Proc. EP 98. New York, pp. 591-600, 1998.

[Shi99] Shi, Y. and Eberhart, R. C., "Empirical study of particle swarm optimisation", Proceedings of the 1999 Congress on Evolutionary Computation. Piscataway, NJ, pp. 1945-1950, 1999.

[Lovbjerg01] Lovbjerg, M.; Rasmussen, T. K.; and Krink, T., "Hybrid particle swarm optimiser with breeding and subpopulations", Proceedings of the third Genetic and Evolutionary Computation Conference (GECCO-2001). 2001.

[Kennedy99] Kennedy, J., "Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance", Proc. Congress on Evolutionary Computation 1999. Piscataway, NJ, pp. 1931-1938, 1999.

[Muhlenbein93] Muhlenbein, H., & Schlierkamp-Voosen, D. (1993). "Predictive models for the breeder genetic algorithm: I. Continuous parameter optimization." Evolutionary Computation, 1 (1), 25-49.