Parametrical Investigation of the Predator Prey Algorithm

Arlindo Silva[1,3], Ana Neves[1,3], Ernesto Costa[2,3]

[1] Escola Superior de Tecnologia, Instituto Politécnico de Castelo Branco,
Av. do Empresário, 6000 Castelo Branco – Portugal
{arlindo, dorian}@est.ipcb.pt

[2] Departamento de Engenharia Informática, Universidade de Coimbra,
Pólo II – Pinhal de Marrocos, 3030 Coimbra – Portugal
ernesto@dei.uc.pt

[3] Centro de Informática e Sistemas da Universidade de Coimbra,
Pólo II - Pinhal de Marrocos, 3030 Coimbra - Portugal

Abstract. In this paper we investigate the influence of parameter choice in the performance of the Predator Prey algorithm. This algorithm is a new variant of the Particle Swarm Optimiser algorithm, where a Predator Prey mechanism was inserted to better control the balance between exploration and exploitation in optimisation problems. We use a higher level Predator Prey Optimiser as a meta-algorithm to search for good parameter combinations for a set of benchmark problems. The results are used to introduce changes in the algorithm and to define guidelines for parameter choice for specific problems.

1 Introduction

The basic particle swarm algorithm was inspired on a metaphor of social interaction between individuals [1]. The result was a population based optimisation algorithm, where individuals, called particles, are represented as vectors of real numbers in a multidimensional space. A second vector is used to represent the particle's velocity. The algorithm searches for optima in the search space by changing the velocity of each particle and, as a result, its trajectory through the search space. The changes in velocity are the result of the attraction of each particle for its previous best position in the multidimensional space, as well as for the best position previously found by all its neighbours. While generally considered a form of evolutionary computation, there is no form of mutation, recombination or even selection in the particle swarm algorithm. Further information on the particle swarm optimiser (PSO), its variants and the underlying cultural model can be found in [2].

It has been noted in previous work [3][4] that, while successful in the optimisation of several difficult benchmark problems, the PSO presents problems in controlling the balance between exploration and exploitation, namely when fine tuning around the optimum is attempted. In [5] we present a new variant of the particle swarm algorithm, called the predator prey optimiser (PPO), where a new particle, the predator, was

introduced. This particle is attracted to the best individual in the population, while all other particles (prey) are repelled by the predator with a strength that increases with proximity. While this mechanism proved to be successful in increasing the performance of the algorithm in several benchmark optimisation problems it also introduced several new parameters. These allowed an increased control over the balance between exploration and exploitation, but combinations of parameters resulting in optimised performance were not known and the parameter space was too large for any form of enumeration. In this work we used a predator prey optimiser as a meta-algorithm in the search of good parametrical combinations for the optimisation of a set of benchmark minimization problems by PPOs using the parameters that were encoded as particles in the swarm of the meta-algorithm.

This was done in order to pursue three main objectives:

- Investigating the usefulness of the PPO as a meta-algorithm for optimising the parameter set of a lower level optimiser.
- Identification of possible changes to the algorithm as a result of regularities in the parameters (e.g. parameters always near 0 may suggest a simplification of the algorithm).
- Identification of good candidates as sets of parameters for optimising functions that present specific problems to the algorithm.

In the next section we describe in more detail the basic particle swarm and the predator prey optimisers. Section 3 is dedicated to the description of the experimental setup, as well as the presentation of the results obtained. In Section 4 we discuss these results relatively to the objectives stated above and draw some final conclusions to this work.

2 The Optimiser Algorithms

2.1 The Swarm Particle 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 [6]; here we assume that every particle is a neighbour to every other particle in the swarm. The general equations for updating the position and velocity for some particle i are the following:

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

In the above formula χ is the constriction coefficient described in [7], 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 a linear decreasing weight. The swarm is usually run for a limit number of iterations or until an error criterion is met.

From (1) we can derive the two most usual ways in which convergence and, as a result, the balance between exploration and exploitation are controlled. [4] uses χ=1 and weight w decreasing linearly from w_{max} to w_{min} during the execution of the algorithm. [7] guaranties convergence by choosing appropriated values for χ and φ=φ_1+ φ_2. w is fixed and equal to 1 in this approach.

2.2 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 swarm intelligence principle behind the algorithm. Other mechanisms could perhaps have been used to the same effect, but it seemed more appropriate to introduce a mechanism that could also be implemented as a distributed behaviour in the swarm. 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:

$$\begin{cases} V_p(t) = \varphi_3(X_g(t-1) - X_p(t-1)) \\ X_p(t) = X_p(t-1) + V_p(t) \end{cases} \quad (2)$$

φ_3 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_3 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. For some particle *i*, if there is no change in the velocity in a dimension *j* the update rules in that dimension still are:

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

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

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

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 Euclidean distance between predator and prey. *D(x)* is an exponential decreasing distance function:

$$D(x) = ae^{-x/b} \quad (5)$$

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), the predator's influence is smaller and usual swarm dynamics take control. The *a* and *b* parameters define the form of the *D* function: *a* represents the maximum amplitude of the predator effect over a prey and *b* allows 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. The constriction coefficient is set to 1.

3 Experimental Results

In this work we used a PPO as a meta-algorithm for finding good parameter sets for a lower level PPO being used to optimise a set of functions commonly utilized in particle swarm [3][4][7] and evolutionary computation [8] as benchmark problems. These functions were chosen because they present different difficulties to the PPO, so it was expected that different parameter sets should be needed to optimise the algorithm performance for each one.

$$
\begin{aligned}
f_1(x) &= \sum_{i=1}^{n-1}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2) \\
f_2(x) &= \sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10) \\
f_3(x) &= \frac{1}{4000}\sum_{i=1}^{n}(x_i-100)^2-\prod_{i=1}^{n}\cos\left(\frac{x_i-100}{\sqrt{i}}\right)+1 \\
f_4(x) &= 418.9829n+\sum_{i=1}^{n}-x_i\sin\left(\sqrt{|x_i|}\right)
\end{aligned}
\tag{6}
$$

In the functions above, x is a real number, n-dimensional vector and x_i is the ith element of that vector. f_1 is the generalized Rosenbrock function, a unimodal function where the optimum is situated near the origin in a long, slightly decreasing, curved valley, where the difference between particles' fitness is very small, thus making the search difficult. f_2 is the generalized Rastrigin function and f_3 the generalized Griewank function, two multimodal functions with many local minima set around the global minima in an unimodal macrostructure. But, while for f_2 the PPO shows difficulties in getting near the optimum, for f_3 the difficulty seems to be the fine tuning stage when the particles are already near the optimum (see [5]). f_4 is the Schwefel function which is also a multimodal function, but is 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 minimisation problems. The search space and initialisation limits for each function are presented in Table 1. The functions were optimized in 30 dimensions with an iteration limit of 2000.

Table 1. Search space and initialisation limits for each function.

Function	Search Space	Init. Limits
f_1	$[-100,100]^n$	$[15,30]^n$
f_2	$[-10,10]^n$	$[2.56,512]^n$
f_3	$[-600,600]^n$	$[300,600]^n$
f_4	$[-500,500]^n$	$[-500,500]^n$

In a first set of experiments the meta-algorithm was ran for 200 iterations or until 50 iterations elapsed without any improvement in the fitness of the best individual. Each particle encoded the eight parameters of a PPO: $[\varphi_1, \varphi_2, \varphi_3, P_f, a, b, w_{min}, w_{max}]$. The search space limits and values for V_{max} for each parameter are presented in Table 2.

Table 2. Search space limits and V_{max} for each parameter.

	φ_1	φ_2	φ_3	P_f	a	b	w_{min}	w_{max}
X_{min}	0	0	0	0	0	0	0	0
X_{max}	5	5	5	0.1	10	10	2	2
V_{max}	2.5	2.5	2.5	0.05	5	5	1	1

The meta-PPO was run with parameters that have been proved in previous work [5] to result in acceptable performance for a wide range of functions: φ_1=2, φ_2=2, φ_3=0.1, P_f=0.002, $a=0.1X_{max}$, $b=0.1X_{max}$, w_{min}=0, w_{max}=0.5.

Each particle (parameter set) was evaluated by running the PPO 30 times with the respective parameters for the function being optimized. The fitness of the particle was defined as the average best value found over the 30 runs. Table 3 presents 3 representative sets of parameters for each function, chosen from 10 executed experiments. The PPO and Fitness columns allow the comparison of the average best particle found by a PPO with standard parameters [5] (over 100 runs) and a PPO using the parameter set found.

Table 3. Typical parameter sets found for each function.

F	PPO	φ_1	φ_2	φ_3	f	a	b	w_{min}	w_{max}	Fitness
f_1	163,9791	2,0622	0,8025	2,1557	0,0324	2,1421	0,0008	0,0595	0,1420	14,2283
		0,4486	2,1009	2,9482	0,0005	4,9887	1,3829	0,2265	0,5820	21,5599
		1,3022	1,8574	0,0780	0,0008	3,6346	3,9220	0,2801	0,3181	8,0003
f_2	10,7441	0,0089	2,4337	0,2065	0,0032	6,7914	1,4256	0,0420	0,1385	2,29E-10
		0,1129	1,7565	2,3885	0,0028	1,3996	0,0332	0,2674	0,3804	4,34E-09
		0,0703	1,2949	1,0450	0,0031	0,8349	4,6922	0,0424	0,4551	9,51E-08
f_3	0,0133	1,7592	0,4997	0,0322	0,0046	7,5328	0,2446	0,2193	0,2478	0,0018
		2,1637	1,6711	2,2793	0,0022	0,5506	3,8235	0,0712	0,7111	0,0054
		1,8012	0,6754	0,5784	0,0052	1,2217	3,3731	0,1839	0,0572	0,0012
f_4	724,7053	0,1753	1,5628	1,8675	0,0092	4,2232	0,1347	0,0474	0,0216	0,0004
		0,0345	0,9674	1,0633	0,0053	4,7001	0,4164	0,0591	0,1124	0,0004
		0,1241	1,2823	1,7609	0,0065	1,7154	3,2174	0,1479	0,0674	0,0004

Even allowing for the increased variance in the average fitness of the best particles found by the PPO using these parameter sets, since only 30 runs were performed against the 100 of the PPO with standard parameters, it seems safe to conclude that they result in an increase of the average quality of the solutions found. This increase is particularly relevant for the Rastrigin and Schwefel functions. It can also be observed that, for the same function, very different sets of parameters result in similar average fitness – this is true not only for the selected parameter sets in Table 3 but also for all the 10 sets found for each function. We may conclude that, while correct choice of parameters is essential for good performance of the PPO – the standard parameters result in a significantly poorer performance than the sets found – there is a wide choice of parameters capable of producing good results for a given function. This seems to suggest that it may be possible to find a common parameter set capable of leading to increased performance in all the functions tested.

Another interesting observation from the above results is the fact that several parameter combinations have values very similar and very small for w_{max} and w_{min}. This seems to indicate that the PPO could obtain the same results without the use of the linear decreasing inertia weight, resulting in a simpler algorithm, with less parameters.

We ran a second set of experiments to investigate the ideas suggested by the results of the first set. The meta-algorithm was executed in the same conditions, using the same encoding and parameters, except that the w_{max} and w_{min} parameters were removed from the encoding and set to 0 in the algorithm, so there was no linear decreasing weight and the previous velocity value had no effect on the one being calculated. The second change was on the evaluation of each particle (parameter set). Each particle was now evaluated by running a PPO with the corresponding parameters 30 times for each of the f_2, f_3 and f_4 functions. The fitness of the particle was set to the addition of the averaged best fitness for the three functions. f_1 was not included since it would bias the search as the best values would be significantly larger for this function than for the remaining three. Due to the computational cost of this experiment it was only repeated 5 times and the parameter sets that were found are presented in Table 4.

Table 4. Parameter sets found in the second group of experiments.

φ_1	φ_2	φ_3	f	a	b	**Fitness**
0,0722	1,8216	0,5473	0,0073	3,1437	7,0544	0,0270
0,2818	1,4278	0,8323	0,0062	7,8576	0,3049	0,0412
0,0533	1,2856	3,9217	0,0042	2,3102	0,1118	0,0325
0,0939	1,2045	0,2901	0,0127	6,8568	3,5659	0,0436
0,0270	1,4228	0,4392	0,0071	3,9820	4,9039	0,0365

The results in Table 4 show that it is indeed possible to find parameter sets capable of leading the PPO to good performance simultaneously in three of our test functions. They also imply that these results can be obtained without the use of the linear decreasing weight and previous velocity value component in the velocity update equation.

The parameter set with best fitness was used to run a PPO 100 times for each function to validate the results obtained in the parameter search, where the high variance associated with the 30 runs for each function would probably have led to optimistic estimates of the average best particle fitness. In Figure 1 the evolution of the average best fitness for a PPO using the best parameter set (PPO2) is presented for each function against the average best fitness in the same conditions for a PPO (PPO1) with standard parameters and a basic PSO. The curves show that, while the PPO2 didn't obtained the results expected from the first set of experiments, the results were still significantly better for the Rastrigin and Schwefel functions that the ones of the PPO with standard parameters and the PSO. For the Griewank function the results were marginally worse for the PPO with the new parameters.

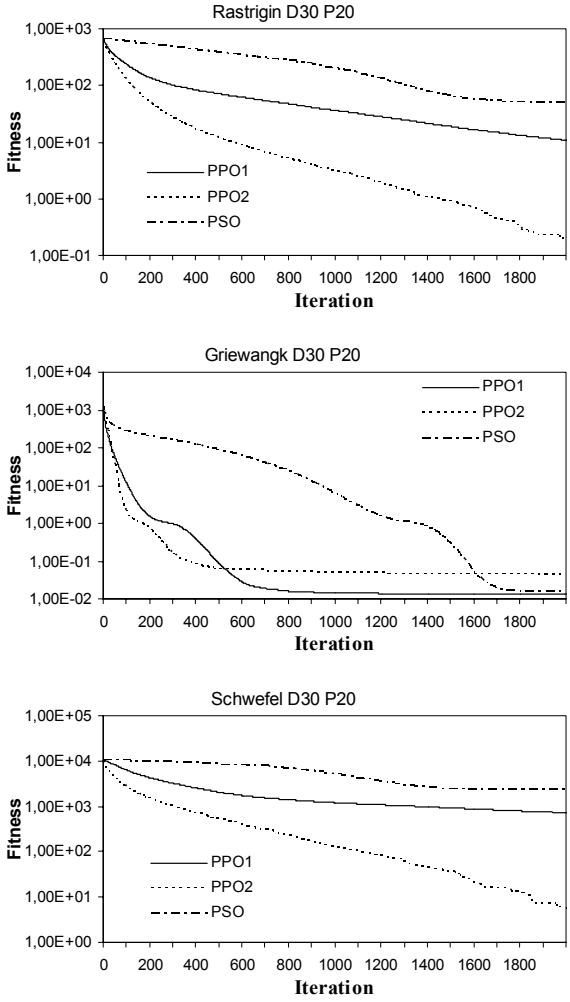

Fig. 1. Graphs illustrating the evolution of average fitness of a PPO with standard parameters (PPO1), a PPO with the new set of parameters (PPO2) and a basic PSO. Fitness is presented in logarithmic scale.

4 Conclusions

In this paper we presented and discussed the results of using a meta-POO to optimize the parameters of a second level PPO evaluated in minimization problems. This was done to pursue the objectives stated in the introduction to this work:

- Investigating the usefulness of the PPO as a meta-algorithm for optimising the parameter set of a lower level optimiser.
- Identification of possible changes to the algorithm as a result of regularities in the parameters.
- Identification of good candidates as sets of parameters for optimising functions that present specific problems to the algorithm.

As for the first objective, we can conclude that the meta-PPO can be used as a tool in optimization to find better parameters for the main optimization algorithm. While this is an expensive procedure in computational terms, it may prove useful when the best possible quality for the final solution is essential.

In relation to the second objective, the first results we obtained suggested that the linearly decreasing weight could be removed from the equations and a second set of experiments where it was not used also showed promising results. While further investigation is needed to confirm that such simplification is desirable, its suggestion by the meta-PPO results shows the meta-algorithm usefulness in identifying interesting changes to the base algorithm. In future work we propose to use the meta-algorithm to search for better predator strategies for the PPO.

The third objective was achieved by proposing parameter sets capable of leading the PPO to better results in the test functions then the ones previously found either by the PPO with standard parameters or the basic PSO. Since the test functions were chosen to present specific problems to the optimizer, we hope the parameter sets found can be useful in the optimization of other functions that pose the same difficulties to the PPO. Generic parameter sets leading to good (but not the best) results in three test functions were also found. These can be used to optimize a new function, if one of the more specific parameter sets does not lead to good results.

Acknowledgements

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

References

1. Kennedy, J. and Eberhart, R. C., “Particle swarm optimisation”, Proc. IEEE International Conference on Neural Networks. Piscataway, NJ, pp. 1942-1948, 1995.

2. Kennedy, J., Eberhart, R. C., and Shi, Y., “Swarm intelligence”, Morgan Kaufmann Publishers, San Francisco. 2001

3. Angeline, P. J., “Evolutionary optimisation versus particle swarm optimisation: philosophy and performance differences”, The Seventh Annual Conf. on Evolutionary Programming. 1998.

4. 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.

5. Silva, A., Neves, A. and Costa E., “Chasing the Swarm: A Predator Prey Approach to Function Optimisation”, Proc. of MENDEL2002 - 8th International Conference on Soft Computing, Brno, Czech Republic, June 5-7, 2002

6. 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.

7. Clerc, M. and Kennedy, J., “The particle swarm-explosion, stability, and convergence in a multidimensional complex space”. IEEE Transactions on Evolutionary Computation, Vol. 6, No. 1, pp. 58-73. 2002.

8. Muhlenbein, H., & Schlierkamp-Voosen, D., “Predictive models for the breeder genetic algorithm: I. Continuous parameter optimisation.” Evolutionary Computation, 1 (1), 25-49.