An Empirical Comparison of Particle Swarm and Predator Prey Optimisation

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 present and discuss the results of experimentally comparing the performance of several variants of the standard swarm particle optimiser and a new approach to swarm based optimisation. The new algorithm, which we call predator prey optimiser, combines the ideas of particle swarm optimisation with a predator prey inspired strategy, which is used to maintain diversity in the swarm and preventing premature convergence to local sub-optima. This algorithm and the most common variants of the particle swarm optimisers are tested in a set of multimodal functions commonly used as benchmark optimisation problems in evolutionary computation.

1 Introduction

The particle swarm algorithm is a new evolutionary algorithm, where the population is now called a swarm and each individual is called a particle [1]. It was inspired by the dynamics of social interactions between individuals, being initially influenced by work in simulation of coordinated flight in flocks of birds. This algorithm, which is frequently called Particle Swarm Optimiser (PSO), has been successfully used as an alternative to other evolutionary techniques in the optimisation of n-dimensional real functions.

The PSO is usually considered a form of evolutionary computation, but it does not make use of mutation or recombination operators and even selection does not exist in an explicit way. Instead, particles move in a coordinated way through the n-dimensional search space towards the function optimum. Their movement is influenced not only by each particle own previous experience, but also by a social compulsion to move towards the best position found by its neighbours. To implement these behaviours, each particle is defined by its position and velocity in the search space. In each iteration, changes resulting from both influences in the particle's trajectory are made to its velocity. The particle's position is then updated accordingly to the calcu-

lated velocity. The PSO, its main variants and the cultural model behind it are extensively discussed in [2].

While the PSO has revealed itself capable of competing with other evolutionary techniques, namely the standard genetic algorithm (GA), it has been noted [3] 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 [3], 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, [4] introduced the use of a linear decreasing inertia weight, reminiscent from the temperature parameter in simulated annealing. In [5] 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. A second approach to controlling convergence in PSO introduces the use of a constriction coefficient [6].

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 experimental results obtained by comparing this model with the inertial weight and the constriction coefficient PSO variants 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 and, in more detail, the new predator-prey optimiser (PPO). Description of the experimental setup and presentation of the results obtained is made in section 3. Finally, section 4 is dedicated to the presentation of some conclusions and objectives for future 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 [7]; 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 [6], 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. [5] uses χ=1 and weight w decreasing linearly from w_{max} to w_{min} during the execution of the algorithm. In [6] convergence is guarantied by choosing appropriated values for χ and $\varphi=\varphi_1+\varphi_2$. 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 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:

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

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. 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:

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

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:

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

$D(x)$ 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

3.1 The Experimental Setup

We tested the performance of the predator-prey model in four non-linear functions, all commonly used as benchmarks for evolutionary algorithms (e.g. [8]). The first two are also frequently used as benchmarks in swarm particle literature [2], [3], [5].

$$
\begin{aligned}
f_1(x) &= \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i) + 10) \\
f_2(x) &= \frac{1}{4000}\sum_{i=1}^{n}(x_i - 100)^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i - 100}{\sqrt{i}}\right) + 1 \\
f_3(x) &= -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e \\
f_4(x) &= 418.9829n + \sum_{i=1}^{n} - x_i \sin\left(\sqrt{|x_i|}\right)
\end{aligned}
\qquad (6)
$$

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 Rastrigin function, f_2 is the generalized Griewank function and f_3 the generalized Ackley function, three multimodal functions with many local minima set around the global minima in an unimodal macrostructure. f_4 is the Schwefel function 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.

In our first set of experiments the predator prey algorithm was used to optimise the above functions. To implement the objective 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, the weight limits were set to 0.5 and 0.0, far from the usual 0.9 and 0.4. φ_1 and φ_2 were set to 2. The a, b, and P_f values were, for each function, empirically and after a small set of preliminary experiments, set to the values presented in table 1. No systematic effort was made to determine optimum values for these parameters.

Table 1. Search space and initialisation limits for each function, as well as parameter values used for the predator prey algorithm.

Function	Search Space	Init. Limits	a	b	P_f
f_1	$[-10,10]^n$	$[2.56,5.12]^n$	$0.1X_{max}$	$10.0/X_{max}$	0.0005
f_2	$[-600,600]^n$	$[300,600]^n$	$0.1X_{max}$	$10.0/X_{max}$	0.002
f_3	$[-30,30]^n$	$[10,20]^n$	$2.0X_{max}$	$10.0/X_{max}$	0.0005
f_4	$[-500,500]^n$	$[-500,500]^n$	$0.1X_{max}$	$10.0/X_{max}$	0.001

We also ran three sets of experiments with variants of the standard particle swarm algorithm. PSOa corresponds to the PSO with the constriction coefficient set to one and a weight w decreasing linearly between 0.9 and 0.4, values recommended in PSO literature [4]. PSOb is similar to the PSOa but w decreases from 0.5 to 0.0 to allow a direct comparison with the PPO, which uses the same values. PSOc is a constricted version, with χ=0.739 and φ=4.1, values used in [9]. In all experimental settings V_{max} was set to the same value as X_{max}.

All experiments were run with functions having 50 dimensions, an iteration limit of 5000 and a population size of 20 particles. Care was taken to initialise the population

in an asymmetric way for all functions except for f4, which, since its structure is not symmetrical around a local minimum situated near the origin, does not benefit from a symmetric initialisation. Table 1 presents the search space and initialisation limits for each of the test functions. For each experimental setting 100 runs of the algorithm were performed.

3.2 Results and Discussion

Table 2 presents 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 best solution found in the limit number of iterations, over the 100 runs. Figure 1 presents graphs illustrating the evolution of average fitness over the run for the four benchmark functions.

Table 2. Confidence intervals at 90% for the average best solutions of each experimental setting over 100 runs.

F	PPO	PSOa	PSOb	PSOc
f_1	5.9357±0.8802	105.8131±4.8350	197.0707±8.1454	282.1091±11.7559
f_2	0.0080±0.0022	0.0136±0.0033	0.0372±0.0145	0.1810±0.0657
f_3	9.1955E-09±2.0367E-09	0.3460±0.1255	2.0451±0.1918	8.0508±0.5647
f_4	797.6098±57.9868	4167.4452±134.8793	6213.0122±156.4541	8242.2676±178.2687

We start by comparing the performances of the three PSO based approaches. It is easy to see from table 2 that the particle swarm optimiser (PSOa) with weight varying from 0.9 to 0.4 is the one that performs better in all experimental settings. From figure 1 can be concluded that this may be caused by the faster convergence of the two other variants in the initial iterations, which causes a decrease in diversity resulting in early sob-optimal convergence for the PSOb and PSOc. It is interesting to compare these results with the ones in [9] where PSOc performed better than PSOa in different experimental settings: of the four functions used only two were common with ours, the functions were optimised in 30 dimensions and, more importantly, the result of each run was the number of iterations the algorithm took to find a solution within a predefined error form the global optimum. Since the error criterion was rather loose, the faster convergence of the PSOc made it perform better than the PSOa in those conditions. But, from our experiments, it seems that when we ask for finer convergence the weight decreasing version of the PSO easily outperforms the constricted version.

In a function to function qualitative analysis, and since the global optimums all have 0.0 fitness, we can say that the PSOa performed well in f_2 and f_3, not very good in the Rastrigin function and rather poorly in f_6 where escaping local minima is harder than in the other functions, since local optima are far apart in the search space. We can add from figure 1 that the PSOa fitness curves seems stabilized at the iteration limit (the same happens with PSOb and PSOc) and doesn't seem probable that an increment in fitness would result from a higher iteration limit.

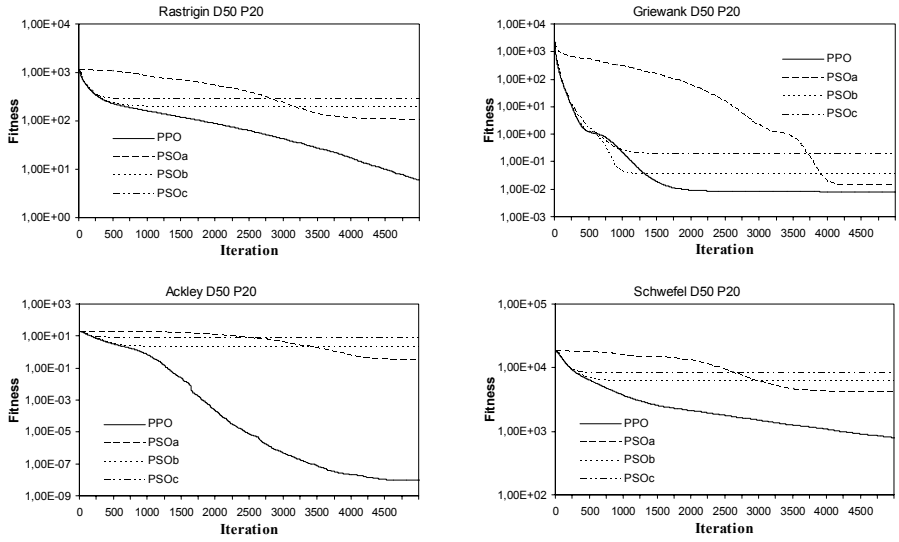

Fig. 1. Graphs illustrating evolution of average fitness of PPO versus PSO approaches for the four test functions. Fitness is presented in logarithmic scale.

The experimental results using the predator prey approach allow us to conclude that the PPO performed significantly better (according to t-tests with $p<0.05$ criterium) than the best PSO approach (PSOa). As we were expecting, the extra diversity introduced by the predator prey mechanism allowed us to use a shorter "cooling schedule" for the linear decreasing weight which resulted, as can be seen in figure 1, in an initial convergence as fast for the PPO as for the PSOb and PSOc. But, while these two algorithms get easily caught in local minima, with their fitness curves stabilising very early in the runs, the PPO keeps getting better solutions until the iteration limit is reached, with exception of f_2. The final results presented in table 2 show that the predator prey mechanism allowed the PPO to find solutions, on average, orders of magnitude better for f_1, f_2 and f_4. For f_2 the solutions are still significantly better in the statistic sense but the difference is not as meaningful as for the other functions. We can conclude that, in the experimental setup we used, the PPO seems to have three advantages over the best PSO tested: significantly better solutions are found, the algorithm converges faster and, since better solutions are still being found at the end of the runs, the PPO can effectively escape local sub-optima. These results apparently support our claim that the predator prey mechanism can constitute an effective way of avoiding premature convergence to local sub-optima in particle swarm based optimisers.

4 Conclusions and Future Work

We presented a new swarm particle based function optimiser, 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 in four benchmark functions against three variants of the standard PSO. The experimental results allow us to conclude that the PPO performed significantly better than the standard PSO in the optimisation 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.

As for the main direction of our future work with the predator prey optimiser, it will probably lead to the development a multi-predator sub-population version of the algorithm.

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., "Parameter selection in particle swarm optimisation" Evolutionary Programming VII: Proc. EP 98. New York, pp. 591-600, 1998.
5. 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.
6. 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.
7. 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.
8. Muhlenbein, H., & Schlierkamp-Voosen, D.. "Predictive models for the breeder genetic algorithm: I. Continuous parameter optimisation." Evolutionary Computation, 1 (1), 25-49.
9. Eberhart, R. C. and Shi, Y. "Comparing inertia weigthts and constriction factors in particle swarm optimization". Proc. CEC 2000 pp. 84-88. San Diego, CA, 2000.