Evolution of Force-Generating Equations for PSO using GP

Cecilia Di Chio, Riccardo Poli, and William B. Langdon

Department of Computer Science, University of Essex, UK
{cdichi, rpoli, wlangdon}@essex.ac.uk

Abstract. We extend our previous research on evolving the physical forces which control particle swarms by considering additional ingredients, such as the velocity of the neighbourhood best and time, and different neighbourhood topologies, namely the global and local ones. We test the evolved extended PSOs (XPS) on various classes of benchmark problems.
We show that evolutionary computation, and in particular genetic programming (GP), can automatically generate new PSO algorithms that outperform standard PSOs designed by people as well as some previously evolved ones.

1 Introduction

Swarm intelligence is an expression used to refer to the emergent collective intelligence of groups of simple agents such as colonies of social insects. Through swarm intelligence it is possible to design intelligent systems (*swarm systems*) in which functions such as control, programming and centralisation have been replaced by more "natural" ones, namely autonomy, emergence and distribution [2]. One of the best-developed techniques of this type is Particle Swarm Optimisation (PSO) [5]. PSOs use a population of "interacting particles" (i.e. candidate solutions) that, controlled by forces, fly over the fitness landscape searching for the optimum.

Since a great variety of improvements to the basic form of PSO has been presented, it is hard to decide what is the best PSO to use to solve a particular class of problems. We propose to solve this problem by evolving the force-generating equations of PSOs. Our aim is not to evolve a PSO which is better than all the others on all possible problems (which is not possible [11]), but instead to evolve different PSOs which could outperform known PSOs on specific classes of interesting problems.

We extend the control law of the forces acting on the particles by adding ingredients, such as the velocity of the neighbourhood best and time. These are believed to be important to allow the adaptation of these forces to the current situation of the swarm. Providing the particles with more information about the overall state of the swarm allows more sophisticated search behaviour.

We also consider two different neighbourhood topologies (global and local) to investigate how particles interaction could affect the evolved forces we use.

We use genetic programming (GP) to automatically generate the equations that generate the forces.

Not all PSOs evolved have a connection with natural swarms, but this is not unusual in computational intelligence. For example, in the field of neural networks, researchers started from biologically inspired neurons but later focused on non-biologically plausible models, such as the back-propagation rule, because of their interesting and useful properties. We expect something like this might happen in the field of PSOs.

In section 2 we provide a brief literature review of the PSO and describe in detail the use of GP to evolve PSOs, along with the results obtained in [9] and in [10]. In section 3 we describe and motivate our extensions of these previous approaches. Section 4 describes our extended PSOs (XPS) and compares them to the performance of human-designed or evolved PSOs on a larger set of benchmark problems than was previously used. In section 5 we briefly restate our findings and list potential future directions for research.

2 Particle Swarm Optimisation

PSO is a method for optimisation of continuous functions, discovered through the simulation of a simplified social model (*particle swarm*) [6]. In PSOs, inspired by flocks of birds and shoals of fish, a number of *particles* are placed in the parameter space of some problem and each evaluates the fitness at its current location. The particles "fly" over the potential solutions, accelerating towards better ones, by combining some aspect of the history of their own fitness values with those of one or more members of the swarm, i.e. their *neighbourhood*, with a velocity determined by their locations and the processed fitness values. The next iteration takes place after all particles have been moved. Eventually the swarm as a whole is likely to move close to the best location.

Every particle has, therefore, two pieces of information available when looking for the optimum, that correspond respectively to *individual learning*, i.e. the personal best fitness value achieved so far, and *social transmission*, i.e. knowledge of how other individuals in its neighbourhood have performed [3].

In this paper, as in most particle swarm implementations, two simple sociometric neighbourhood topologies have been used, namely the *global* neighbourhood – every particle is influenced by the performance of the entire swarm– and the *local* neighbourhood – each particle is influenced only by itself and n closest neighbours (for the purpose of this paper, $n = 2$; see figure 1).

In the simplest version of PSO, each particle is accelerated by two elastic forces. One attracts it to the best location found so far by the particle itself and the other to the best location found so far by the members of the neighbourhood. The magnitude of the force is randomly chosen at each time step. The equation controlling the particles is of the form

$$a_i = F(x_i, x_{s_i}, x_{p_i}, v_i) \tag{1}$$

and more precisely, for the simple PSO

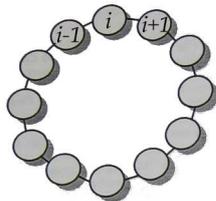$$a_i = (x_{s_i} - x_i)\phi_1 R_1 + (x_{p_i} - x_i)\phi_2 R_2 \tag{2}$$

Fig. 1. Local neighbourhood with $n = 2$. Particles $i-1$ and $i+1$ are particle i's closest neighbours

where:

- a_i is the ith component of the acceleration vector,
- x_i is the ith component of the particle's current location,
- x_{s_i} is the ith component of the best point visited by the particles in the neighbourhood (global or local),
- x_{p_i} is the ith component of a particle's personal best,
- v_i is the ith component of the particle's velocity vector,
- R_1 and R_2 are two independent random variables uniformly distributed in $[0, 1]$,
- ϕ_1 and ϕ_2 are two learning rates which control respectively the proportion of social transmission and individual learning in the swarm.

The velocity v of a particle and its position x are updated every time step using the equations:

$$v_i(t) = v_i(t-1) + a_i \qquad\qquad x_i(t) = x_i(t-1) + v_i(t)$$

As this system can lead the particles to become unstable, with their speed increasing without control, the standard technique to avoid this happening is to bound velocities so that $v_i \in [-V_{max}, +V_{max}]$.

In [4], Clerc and Kennedy developed a comprehensive 5-dimensional mathematical analysis of the basic PSO system. A particularly important contribution of that work was the use and analysis of a modified update rule:

$$v_i(t) = \kappa(v_i(t-1) + a_i) \tag{3}$$

where the constant κ, the "constriction coefficient", if correctly chosen, guarantees the stability of the PSO without the need to bound velocities.

Further explorations of physics-based effects in the swarm include Blackwell's charged particles [1] and Poli and Stephens' particles sliding over the fitness landscape [8].

Extending Particle Swarms

Genetic programming (GP) performs the evolution of programs, represented in the form of a tree, with the use of specialised genetic operators such as crossover

and mutation [7]. The idea in [9] was to interpret the function F in Equation 1 as a program to be evolved by the GP.

A limitation of the standard PSO, and of our first approach as well, was that the information about the global state of the swarm was limited to the x_{s_i} term. In [10] we extended the definition of the force by adding the *centre of mass* of the swarm x_{c_i} and the average distance of the particles from the centre of mass (i.e. the *dispersion* of the swarm around the centre, d), obtaining

$$a_i = F(x_i, x_{s_i}, x_{p_i}, v_i, x_{c_i}, d) \tag{4}$$

We trained the GP on problems taken from the following classes of functions:

City-Block sphere - unimodal with global optimum at $x = (g_1, \cdots, g_N)$, with $f(x) = 0$.

Generalised Rastrigin - multimodal with global optimum at $x = (g_1, \cdots, g_N)$, with $f(x) = 0$, and many regularly distributed local optima.

The g_i values were chosen uniformly at random in the range $[-G, G]$ with $G = 1.0$ and $G = 2.0$, the dimensions of the problems have been set to $N = 2$ and $N = 10$. The best XPSOs evolved were:

psog3 - evolved when the training set was the Rastrigin function and $G = 1.0$ and therefore expected to perform well on highly multimodal objective functions (see [9] for an explanation):

$$F = R_1(x_{s_i} - x_i) - 0.75R_2R_1x_ix_{s_i}^2 - 0.25R_3R_2R_1x_ix_{s_i}$$

psodisp2 - completely deterministic function (i.e. no random elements) evolved when the training set was the City-block sphere function and $G = 2.0$ (see [10]):

$$F = (x_{s_i} - x_i) + (x_{p_i} - x_i) - dx_i$$

psocd1 - 100% social (i.e. no use of information about particle own performance) XPSO evolved when the training set was the City-block sphere functions and $G = 1.0$ (see [10]):

$$F = (x_{s_i} - x_i) - \frac{R^2}{d}v_i$$

On the City-block problem class, the XPSs **psog3** and **psocd1** have better performances than almost all the others. On the Rastrigin class of problems **psog3** and **psodisp2** are the best. Since **psog3** has been evolved on Rastrigin but performs well on City-block problems and, conversely, **psodisp2** has been evolved on City-block but has good performances on Rastrigin problems, **psog3** and **psodisp2** are both good all-rounders.

3 New ingredients

In this paper we have added to the equation of the force the velocity of the best particle in the neighbourhood of a particle v_{s_i} and the normalised time t (i.e.

current generation divided by the total number of generations allowed), evolved XPSs and tested them with different neighbourhood topologies, namely global and local with $n = 2$ neighbours, and verified the validity of the solution proposed here and in the previous papers with a comprehensive set of test functions, namely: Ackley, Griewangk, Rastrigin, Rosenbrock (2nd De Jong's function) and Sphere (1st De Jong's function).

The equation of the force controlling a particle now is:

$$a_i = F(x_i, x_{s_i}, x_{p_i}, v_i, v_{s_i}, x_{c_i}, d, t) \tag{5}$$

We are interested in the velocity v_{s_i} because we want to see if it may be beneficial for particles to try and imitate the full search behaviour of the neighbourhood best (rather than just its current position). The time ingredient has been added to see if and when it is beneficial for particles to change their behaviour as the search progresses.

The training set consists of problems taken from the following function classes:

Sphere[1] - unimodal with one global optimum at $x = (g_1, \cdots, g_N)$, with $f(x) = 0$.

Generalised Rastrigin - see section 2

Table 1 summarises the settings needed by the GP to evolve the equations of the forces. Adding elements to the terminal set does not have any impact on the computational time.

After evolution (training phase), we have tested the forces discovered by GP using a much larger set of benchmark functions:

Generalised Ackley - multimodal with one global optimum at $x = (g_1, \cdots, g_N)$, with $f(x) = 0$, and many local optima.
Griewangk - multimodal with one global optimum at $x = (g_1, \cdots, g_N)$, with $f(x) = 0$, and many regularly distributed local optima.
Generalised Rastrigin - see section 2
Rosenbrock - unimodal with one global optimum at $x = (g_1 + 1, \cdots, g_N + 1)$, with $f(x) = 0$.
Sphere - see above

In the testing phase, done to see how robust the PSOs evolved by GP were, we have generated 30 random problems taken from the benchmark function classes, with dimensions $N = 2$, 10 and 30 and values of g_i chosen uniformly in the range $[-G, G]$, with $G = 1.0$ and $G = 2.0$. For each instance of the problem, we have run 30 independent runs of each PSO.

While in the training phase we were looking for a fitness function which "suggested" to the GP how to evolve more efficient force equations, in the testing stage we are more concerned with the ability of the XPS as a whole to find the global optimum at the end of the PSO iterations, i.e. how close the swarm best gets to it. Therefore, we have used as a performance measure the sum of the distances between the swarm best and the global optimum, i.e. $\sum_i |x_{s_i} - g_i|$.

---

[1] The Sphere function we use here is different from the City-block sphere used in previous research.

Table 1. Koza style tableau, showing parameter settings for the evolution of force-generating equations for PSO

Objective	Evolve $F(x_i, x_{s_i}, x_{p_i}, v_i, v_{s_i}, x_{c_i}, d, t)$ over 10 random problems from the Sphere or Rastrigin function classes; selected problems have two dimensions ($N = 2$) and values for the global optimum randomly chosen in $x_1 = -1.0, \cdots, +1.0$, $x_2 = -1.0, \cdots, +1.0$
Terminal set	$x_i, x_{s_i}, x_{p_i}, v_i, v_{s_i}, x_{c_i}, d, t, -1.0, -0.5, 0.5, 1.0$ and a random number generator R which returns numbers uniformly distributed within $[-1, 1]$
Function set	$+, -, \times$ and the protected division DIV^2
Fitness cases	10 particles with initial position chosen uniformly at random in the interval $x_1 = -5.0, \cdots, +5.0$, $x_2 = -5.0, \cdots, +5.0$ and initial velocity set to 0; each PSO has been run for 30 iterations on each problem; the velocity of each particle has been updated using Clerc's update rule (Equation 3) with $\kappa = 0.7$ and the components of the velocity vector constrained within $[-2.0, 2.0]$
Raw fitness	$\sum_x \sum_i \|x_i - g_i\|$, i.e. the sum of the distances between each particle and the global optimum. This has been done to encourage the convergence of the whole swarm at the optimum in order for the GP to evolve better forces. Evolution of simpler forces has been guaranteed by the use of a mild parsimony pressure coefficient
Population size	1000 forces
Initialisation method	Random
Simulation time	100 generations
Crossover probability	90% standard sub-tree crossover (with uniform random selection of crossover points)
Mutation probability	10% point mutation with a 2% chance of mutation per node
Initial program length	6 levels, the root being at level 0
Selection scheme	Steady state binary tournaments for parent selection and binary negative tournaments to decide who to remove from the population
Termination criteria	Automatically at generation 100

4 Results

We have run the tests on 11 new XPSs[3] (6 with global and 5 with local neighbourhood), as well as on the three XPSs evolved in the previous experiment (see section 2) and some hand-designed PSOs.

For completeness and also to test the performance of previously evolved PSOs on local neighbourhoods, we decided to interpret the x_{s_i} term as the best of the

---

[2] If $|y| <= 0.001$ DIV$(x, y) = x$ else DIV$(x, y) = x/y$.

[3] The XPSs chosen for the testing have been selected, according to efficiency and simplicity criteria, from a pool of many candidates PSOs evolved by the GP during the training phase.

neighbourhood, either global or local, even for the XPSs from previous work, which were evolved using only a global neighbourhood topology.

The three human-designed PSOs used in the test are[4]:

pso - version of the standard PSO with $\phi_1 = \phi_2 = 1.0$
psod1 - 100% social version of the standard PSO with no random coefficients and $\phi_1 = 1$ and $\phi_2 = 0$
psor1 - 100% social version of the standard PSO with random coefficients and $\phi_1 = 1$ and $\phi_2 = 0$

Among the 11 newly evolved XPSs tested, we present here only the best performing ones (3 for the global and 2 for the local neighbourhood). These are also the ones with the "simplest" equation, i.e. those whose forces have a connection with natural swarms and/or for which it is easier to give a physical interpretation.

For each of the five benchmark functions, table 2 and table 3 summarise (for global and local neighbourhoods, respectively) the two best results obtained in testing these newly evolved PSOs against the existing ones.

Global neighbourhood

The best XPSs evolved by the GP with a global neighbourhood are:

psosbdt - evolved on the Rastrigin function, with dispersion and time in the terminal set (in addition to the standard PSO ingredients)

$$F = \frac{1}{t}(x_{s_i}t - x_i) + d$$

psosbtime - evolved on the Rastrigin function, with only time in the terminal set

$$F = (t + 1)(x_{s_i}t - x_i)$$

psosbtv - evolved on the Sphere function, with both time and neighbourhood best's velocity in the terminal set, but no dispersion or swarm centre

$$F = (x_{s_i}t - x_i) - 0.5v_i(R + t^2)$$

Local neighbourhood

The best XPSs evolved by the GP with a local neighbourhood are:

psolbdt - evolved on the Rastrigin function, with dispersion and time in the terminal set

$$F = (x_{s_i}t - x_i)$$

psolbtv - evolved on the Sphere function, with time and neighbourhood best's velocity in the terminal set

$$F = (x_{s_i} - x_i) - v_i t(1 + 3v_i t(x_{s_i} + v_i))$$

---

[4] See [9] for an exhaustive explanation.

Table 2. Mean and standard deviation (in brackets) over 30 runs of normalised distance between swarm best found by each PSO and the optimum for each function which uses a global neighbourhood. Best results in bold

Global neighbourhood		$N = 2$ $G = 1.0$	$G = 2.0$	$N = 10$ $G = 1.0$	$G = 2.0$	$N = 30$ $G = 1.0$	$G = 2.0$
Ackley	psosbdt	0.3101	0.4268	0.3891	0.5087	**0.4740**	0.5712
		(0.0730)	(0.1097)	(0.0503)	(0.0527)	(0.0384)	(0.0529)
	psosbtime	**0.2420**	**0.3899**	**0.3843**	**0.4901**	0.4755	**0.5522**
		(0.0629)	(0.1017)	(0.0347)	(0.0545)	(0.0510)	(0.0459)
Griewangk	psosbg3	**0.0170**	0.2605	**0.0685**	**0.1915**	**0.1299**	**0.2709**
		(0.0202)	(0.2445)	(0.0184)	(0.0512)	(0.0115)	(0.0405)
	psosbtime	0.0598	**0.2210**	0.1212	0.2583	0.1543	0.3089
		(0.0343)	(0.1687)	(0.0283)	(0.0514)	(0.0153)	(0.0420)
Rastrigin	psosbg3	**0.0997**	**0.1536**	0.1851	**0.3056**	0.1968	**0.3357**
		(0.0235)	(0.0488)	(0.0179)	(0.0414)	(0.0139)	(0.0235)
	psosbtime	0.1130	0.2145	**0.1777**	0.3171	**0.1856**	0.3370
		(0.0512)	(0.0947)	(0.0291)	(0.0428)	(0.0146)	(0.0276)
Rosenbrock	psosbg3	0.2179	0.2233	**0.0797**	**0.1896**	**0.1320**	**0.2806**
		(0.0430)	(0.0856)	(0.0200)	(0.0453)	(0.0122)	(0.0269)
	psosbtime	**0.1387**	**0.1931**	0.1236	0.2599	0.1467	0.3172
		(0.0582)	(0.0992)	(0.0288)	(0.0535)	(0.0158)	(0.0330)
Sphere	psosbg3	0.0036	0.0136	**0.0615**	**0.1451**	**0.1191**	**0.2508**
		(0.0023)	(0.0099)	(0.0102)	(0.0269)	(0.0099)	(0.0269)
	psosbtv	**0.0000**	**0.0000**	0.1424	0.1662	0.2824	0.3487
		(0.0000)	(0.0000)	(0.0074)	(0.0099)	(0.0078)	(0.0158)

4.1 Analysis

The behaviour of the two best XPSs for both global and local neighbourhoods, namely **psosbtime** and **psolbdt**, is very similar.

From the temporal analysis in Table 4, we can conclude that:

- both the forces are deterministic (i.e. no random terms) and do not use the personal best x_{p_i};
- at the beginning of the run (**t=0**), all particles are attracted towards the area where the optima typically are (around the origin);
- as the run progresses, the neighbourhood best becomes progressively the attractor for the particles. Eventually ($\mathbf{t \simeq 1}$), the bias towards the origin disappears.

For **psosbtime**, we give a possible explanation as to why GP has evolved that particular force. Recalling from section 2 the update rule modified with the constriction factor (eq. 3), we can rewrite the equation as

$$v_i(t) = \kappa v_i(t - 1) + \kappa F$$
$$= v_i(t - 1) - (1 - \kappa)v_i(t - 1) + \kappa F$$

Table 3. Mean and standard deviation (in brackets) over 30 runs of normalised distance between swarm best found by each PSO and the optimum for each function which uses a local neighbourhood. Best results in bold

Local neighbourhood		$N = 2$ $G = 1.0$	$N = 2$ $G = 2.0$	$N = 10$ $G = 1.0$	$N = 10$ $G = 2.0$	$N = 30$ $G = 1.0$	$N = 30$ $G = 2.0$
Ackley	psolbg3	0.3045	0.4241	0.4514	0.5535	0.5560	0.6368
		(0.0643)	(0.1032)	(0.0588)	(0.0791)	(0.0531)	(0.0562)
	psolbdt	**0.2670**	**0.4062**	**0.4252**	**0.5209**	**0.5442**	**0.6256**
		(0.0602)	(0.1177)	(0.0375)	(0.0535)	(0.0425)	(0.0396)
Griewangk	psolbg3	**0.0249**	**0.1103**	**0.0971**	**0.2231**	**0.1460**	**0.2895**
		(0.0112)	(0.1016)	(0.0139)	(0.0453)	(0.0131)	(0.0286)
	psolbdt	0.0622	0.1681	0.1300	0.2737	0.1523	0.3156
		(0.0283)	(0.1046)	(0.0215)	(0.0508)	(0.0149)	(0.0304)
Rastrigin	psolbg3	**0.1230**	**0.1894**	0.1975	**0.3223**	0.2021	0.3436
		(0.0320)	(0.0654)	(0.0190)	(0.0328)	(0.0126)	(0.0295)
	psolbdt	0.1401	0.2297	**0.1876**	0.3326	**0.1828**	**0.3400**
		(0.0638)	(0.1030)	(0.0265)	(0.0394)	(0.0154)	(0.0308)
Rosenbrock	psolbg3	0.1808	0.1741	**0.1081**	**0.2341**	**0.1498**	**0.2922**
		(0.0612)	(0.0785)	(0.0172)	(0.0374)	(0.0095)	(0.0338)
	psolbdt	**0.1538**	**0.1697**	0.1327	0.2792	0.1525	0.3124
		(0.0636)	(0.0970)	(0.0232)	(0.0415)	(0.0127)	(0.0376)
Sphere	psolbg3	0.0176	0.0671	**0.0882**	**0.1936**	**0.1407**	**0.2762**
		(0.0077)	(0.0417)	(0.0104)	(0.0350)	(0.0116)	(0.0264)
	psolbtv	**0.0004**	**0.0129**	0.1039	0.2086	0.1547	0.2940
		(0.0002)	(0.0155)	(0.0119)	(0.0351)	(0.0112)	(0.0256)

Table 4. Behaviour of **psosbtime** and **psolbdt** at different times

	psosbtime	psolbdt
t=0	$F = -x_i$	$F = -x_i$
t=0.5	$F = 1.5(0.5(x_{s_i} - x_i) - 0.5x_i)$	$F = 0.5(x_{s_i} - x_i) - \frac{x_i}{2}$
t=1	$F = 2(x_{s_i} - x_i)$	$F = (x_{s_i} - x_i)$

We can consider the second term $-(1-\kappa)v_i(t-1)$ as friction, while the coefficient multiplying F (the constriction factor, κ) can be interpreted as the *inverse* of the mass of the particles. In all our experiments we compose our force generating equations with this update rule (with $\kappa = 0.7$). So, the force is first scaled by κ and then added to the friction component $-(1 - \kappa)v_i(t - 1)$. In the case of **psosbtime**, a time varying scaling factor $(1+t)$ has been evolved. This effectively means that at different times the particles have different masses.

At time **t=0**, the mass is $\kappa^{-1} \approx 1.4$. However, as time progresses, first the effects of the constriction coefficient are completely removed (at **t=0.5** the mass is $\frac{1}{\kappa(1+t)} \approx 1$) and then the mass is eventually reduced to approximately 0.7 at **t=1**.

This analysis gives a qualitative explanation of the behaviour of **psosbtime** (being quantitatively exact only if the swarm optimum is in the origin).

5   Conclusion

We have extended our previous research [9] and [10] on exploring extensions for PSO by adding some meaningful ingredients to the equations that generate the forces governing the particles in the swarm, namely the velocity of the neighbourhood best and time.

The addition of the time term has confirmed that it may be beneficial for the particles to change their behaviour as the search progresses. On the other hand, information about the velocity of the neighbourhood best does not seem to provide any improvements. Also, changing the neighbourhood topology does not seem to significantly affect the search.

Future investigations will focus on considering other ingredients, as well as on a deeper study of the good XPSs that have been evolved in this and past research. We also want to explore the effects of using different performance measures. Last but not least, we intend to apply this approach to a variety of real-world problems.

References

1. T.M. Blackwell and P.J. Bentley, *Dynamic Search with Charged Swarms*, Proceedings of the Genetic and Evolutionary Computation Conference, 19–26, 2002.
2. E. Bonabeau, M. Dorigo and G. Theraulaz, *Swarm Intelligence: from Natural to Artificial Systems*, Oxford University Press, 1999.
3. R. Boyd and P.J. Richerson, *Culture and the Evolutionary Process*, The University of Chicago Press, 1985.
4. M. Clerc and J. Kennedy, *The Particle Swarm - Explosion, Stability, and Convergence in a Multidimensional Complex Space*, IEEE Transactions Evolutionary Computation, 6, 1, 58-73, 2002.
5. J. Kennedy and R.C. Eberhart, *Swarm Intelligence*, Morgan Kaufmann Publishers, 2001.
6. J. Kennedy and R.C. Eberhart, *Particle Swarm Optimization*, Proceedings of IEEE International Conference on Neural Networks, 1942-1948, 1995.
7. W.B. Langdon and R. Poli, *Foundations of Genetic Programming*, Springer-Verlag, 2002.
8. R. Poli and C.R. Stephens, *Constrained Molecular Dynamics as a Search and Optimization Tool*, European Conference on Genetic Programming, 150-161, 2004.
9. R. Poli, W.B. Langdon and O. Holland, *Extending Particle Swarm Optimisation via Genetic Programming*, European Conference on Genetic Programming, 291-300, 2005.
10. R. Poli, C. Di Chio and W.B. Langdon., *Exploring Extended Particle Swarm: a Genetic Programming Approach*, Proceedings of the Genetic and Evolutionary Computation Conference, 169-176, 2005.
11. D. Wolpert and W.G. Macready, *No Free Lunch Theorems for Optimization*, IEEE Transactions on Evolutionary Computation, 1, 1, 67-82, 1997.