Egocentric Particle Swarm Optimization

Foundations of Evolutionary Computation
Mandatory Project 1

Magnus Erik Hvass Pedersen (971055)
February 2005, Daimi, University of Aarhus

1 Introduction

The purpose of this report is to verify attendance of the author to the *Foundations of Evolutionary Computation* course at the department of computer science, University of Aarhus.

The reader is assumed to be familiar with the problem description as well as the course litterature. First the concept of function optimization is described, along with the basic *Particle Swarm Optimization* (PSO) formulae. Then a simplification of this method is given, whose parameters are found by socalled *meta*-optimization, and finally, experiments are conducted. These experiments are slightly different from those suggested in the project-description, so as to easen the performance comparison between the Ego-PSO and the PSO.

2 Function Optimization

Consider a set X and an associated fitness function:

$$f : X \to \mathbb{R}$$

To minimize f we must find $x \in X$ so that $\forall x' \in X : f(x) \leq f(x')$. Such a point x is known as a global minimum, and the best known thus far in an optimizational run, is denoted $\vec{gbest}$ with fitness $gbest$. In this paper, our concern is minimization of functions over n-dimensional real-valued domains: $X \subset \mathbb{R}^n$.

3 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is described in [1], and works by letting the socalled particles try and improve their positions in the search-space by exploring close to their own previous best solution, as well as the best previous solution of their peers. The particles are initially placed at random positions, moving in randomly chosen directions. The direction of an particle is then gradually changed so it will start to move towards the best known positions, searching in their vicinity and hopefully discovering even better positions.

3.1 Position & Velocity Formulae

The position of an particle is denoted $\vec{x}$, and its velocity is denoted $\vec{v}$. The update formula for the velocity is then:

$$\vec{v} \leftarrow \omega\,\vec{v} + \phi_1\,r_1\,(\vec{p} - \vec{x}) + \phi_2\,r_2\,(\vec{g} - \vec{x}) \tag{1}$$

and then adding this to the particle's old position, results in its new position:

$$\vec{x} \leftarrow \vec{x} + \vec{v} \tag{2}$$

The inertia weight $\omega \in \mathbb{R}$ determines the persistence of the previous velocity, and hence the particle's tendency to follow its initial path. The particle's own best position up until now is given by $\vec{p}$, and its neighbours' best is $\vec{g}$. These are weighted by the random $r_1, r_2 \sim U(0,1)$ and scaled by $\phi_1, \phi_2 \in \mathbb{R}$ respectively.

3.2 Parameters

In [2], the best choice of PSO parameters for optimizing the first five benchmark functions from section 5, were found to be:

$$\phi_1 = 4.11392,\ \phi_2 = 0.399509,\ \omega = 0.575443,\ |S| = 165 \tag{3}$$

where $|S|$ is the number of particles in the swarm.

4 Egocentric PSO

Consider ϕ_1 and ϕ_2 from Eq.(3). Clearly there is a large bias towards ϕ_1 which means the influence in Eq.(1) of a particle's own previous best solution $\vec{p}$, is much larger than that of its neighbour's previous best, $\vec{g}$.

The idea in this study, is therefore to remove $\vec{g}$ from the velocity update, and use meta-optimization from [2], to find the parameters for this variant of the PSO method. The variant is dubbed *Egocentric* PSO (Ego-PSO), for which the velocity update in Eq.(1), is merely simplified to be the following:

$$\vec{v} \leftarrow \omega\,\vec{v} + \phi_1\,r_1\,(\vec{p} - \vec{x}) \tag{4}$$

with the variables defined as above, and the velocity is being added to the particle's current position $\vec{x}$ as usual (see Eq.(2)).

As with the PSO, the velocity for a particle in the Ego-PSO, is bounded to the dynamic range of the search-space, meaning that a particle can move from one end of the search-space to the other, in a single step.

4.1 No Inter-Particle Communication

Communication between particles in the original PSO method, is done through $\vec{g}$, the best known position of the particle's neighbours. Removing this from the velocity update formula, therefore also removes the particle's communication with each other.

4.2 Meta-Optimization of Parameters

Since the particles do not communicate with each other in the Ego-PSO, then the swarm-size is irrelevant, and is therefore fixed to 20, even though a single particle would suffice. The other parameters for the Ego-PSO are found by socalled *meta*-optimization, in which one heuristical optimization method is made to find the best choice of parameters for another optimization method, that in turn work to solve a given set of problems. This technique is described in [2], and the optimization of the parameters of the Ego-PSO follow that of the PSO's parameters in [2, Section 8.1], however with the DE meta-swarm using the parameters in [2, Eq.(9)], and only performing a single meta-optimizational run on the Ego-PSO (due to time-constraints on the hand-in of this project).

The parameters found for the Ego-PSO by means of meta-optimization, were as follows:

$$\phi_1 = 2.56375,\ \omega = 0.863413 \tag{5}$$

and again with $|S| = 20$ particles in the swarm.

5 Benchmark Functions

To compare the performance of one optimization method against another, each of the following benchmark problems is optimized 50 times with a suitable number of particle-steps[1] (here, 4e4 times the dimensionality of the problem), and the average *gbest* is taken.

The functions that are to be minimized, are all continuous, and with the exception of Rosenbrock, have their global minimum in $\vec{0}$ with $f(\vec{0}) = 0$. The global minimum for Rosenbrock is in $\vec{1}$, and again with a fitness of zero. Note that also the Schaffer-function is generalized to n dimensions.

- **Sphere**:

$$f_1(\vec{x}) = \sum_{i=1}^{n} x_i^2,\ |x_i| \leq 100$$

- **Griewank**:

$$f_2(\vec{x}) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1,\ |x_i| \leq 600$$

- **Rastrigin**:

$$f_3(\vec{x}) = \sum_{i=1}^{n} \left(x_i^2 + 10 - 10 \cdot \cos(2\pi x_i)\right),\ |x_i| \leq 5.12$$

[1] A particle-step is the updating of position and recalculation of fitness for one particle.

- **Ackley**:

$$f_4(\vec{x}) = e + 20 - 20 \cdot \exp\left(-0.2 \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right),\ |x_i| \leq 30$$

- **Rosenbrock**:

$$f_5(\vec{x}) = \sum_{i=1}^{n-1}\left(100 \cdot (x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right),\ |x_i| \leq 100$$

- **Schaffer**:

$$f_6(\vec{x}) = 0.5 + \frac{\sin^2\left(\sqrt{f_1(\vec{x})}\right) - 0.5}{(1 + 0.001 \cdot f_1(\vec{x}))^2},\ |x_i| \leq 100$$

Although particles may initially be placed in the full valid range of a function, the symmetry of e.g. the Sphere-function, suggests that particles should initially be placed asymmetrically in the search-space, so as to increase the difficulty of optimizing the problem. Such initialization ranges are given in table 1.

Function	Minimum	Maximum
Sphere	50	100
Griewank	300	600
Rastrigin	2.56	5.12
Ackley	15	30
Rosenbrock	15	30
Schaffer	50	100

Table 1: Asymmetrical initialization ranges for the benchmark functions.

6 Implementation

Implementation is carried out in *MS Visual C++ .NET* and compiles to an *MS Windows* executable. The abstract base-class for a particle (i.e. its position in the search-space) is `LVector`, which is derived into `LVectorReal` that implements a real-valued n-dimensional search-space, which is then subclassed into six classes, one for each of the benchmark problems above.

The base-class for a swarm is `LSwarm`, which is sub-classed into `LSwarmEgoPSO` that implements the actual Ego-PSO. The entry-point is found in `ec_egopso.cpp`, which also has some IO-routines for obtaining various settings and parameters from the user, as well as routines for creating the objects that will perform the optimization.

7 Experimental Results

The experiments on the above benchmark functions using the Ego-PSO, are summarized in table 2. Compare this to the results of the original PSO as given in table 3 (reprinted from [2], with the exception of the Schaffer benchmarks).

As can be deduced from the average *gbest*-values and their standard deviations, the Ego-PSO is capable of finding near-optimal fitnesses for about half of the benchmark functions, although its performance is overall quite irregular. The results are much worse than those for the PSO in table 2, which underlines the influential importance of a particle's neighbours.

Problem	20 dim.	50 dim.	100 dim.
Sphere	9292.8 (10310)	25982.1 (21812.6)	53440.8 (32677)
Griewank	117.4 (100.9)	229.17 (169.34)	400.89 (327.42)
Rastrigin	123.92 (53.45)	450.89 (96.55)	1027.41 (184.32)
Ackley	19.92 (0.009)	19.94 (0.002)	19.95 (0.001)
Rosenbrock	14.18 (7.93)	47.77 (6.82)	90.59 (26.71)
Schaffer	0.28 (0.24)	0.35 (0.23)	0.40 (0.20)

Table 2: Ego-PSO with the parameters from Eq.(5), and with asymmetrical initialization ranges of its particles. Table shows the average *gbest* of 50 runs, where the number of particle-steps for each run is the dimensionality times 40000. Numbers in parenthesis are the standard deviations.

Problem	20 dim.	50 dim.	100 dim.
Sphere	1.12e-7 (1.53e-7)	1.92e-7 (3.06e-7)	1.43e-7 (1.80e-7)
Griewank	8.36e-7 (1.24e-6)	4.32e-7 (7.74e-7)	2.90e-7 (4.49e-7)
Rastrigin	1.49 (5.91)	8.95 (13.86)	11.94 (23.98)
Ackley	4.69e-5 (2.53e-5)	4.95e-5 (3.45e-5)	3.81e-5 (2.75e-5)
Rosenbrock	1.09 (3.74)	7.24 (16.58)	7.51 (25.46)
Schaffer	7.04e-3 (4.22e-3)	3.60e-3 (4.61e-3)	2.20e-3 (4.00e-3)

Table 3: PSO with the parameters from Eq.(3), and with asymmetrical initialization ranges of its particles. Table shows the average *gbest* of 50 runs, where the number of particle-steps for each run is the dimensionality times 40000. Numbers in parenthesis are the standard deviations.

8 Conclusion

A curious simplification of *Particle Swarm Optimization* (PSO) was presented, in which the particles only depend on their own velocities as well as previous best positions, thus disregarding the previous best positions of their peers. This variant was dubbed *Egocentric* PSO (Ego-PSO).

Parameters were found for the Ego-PSO by employing *meta*-optimization on a handful of common benchmark problems. Experiments were then conducted

with these parameters, on the same benchmark problems, albeit also in higher dimensions, and also on the generalized Schaffer function.

Although the Ego-PSO was capable of getting close to the global optimum for some of these problems, its general performance was nowhere near that of the basic PSO. Therefore, communication between particles appears to be a significant contribution to the PSO's efficiency, even though [2] found it to be much less influential than the attraction of a particle's own previous best position in the search-space.

References

[1] R. C. Eberhart and Y. Shi. Particle Swarm Optimization: Developments, Applications and Resources. Proceedings of the Congress on Evolutionary Computation, vol. 1, pp. 81-86, 2001.

[2] M. E. H. Pedersen. Efficient Meta-Optimization. January 2005.