Good Parameters for Particle Swarm Optimization

By
Magnus Erik Hvass Pedersen
Hvass Laboratories
Technical Report no. HL1001
2010

Abstract

The general purpose optimization method known as Particle Swarm Optimization (PSO) has a number of parameters that determine its behaviour and efficacy in optimizing a given problem. This paper gives a list of good choices of parameters for various optimization scenarios which should help the practitioner achieve better results with little effort.

Keywords: Numerical optimization, particle swarm, parameters.

1 Introduction

The general purpose optimization method known as Particle Swarm Optimization (PSO) is due to Kennedy, Eberhart and Shi [1] [2] and works by maintaining a swarm of particles that move around in the search-space influenced by the improvements discovered by the other particles.

The advantage of using an optimization method such as PSO is that it does not use the gradient of the problem to be optimized, so the method can be readily employed for a host of optimization problems. This is especially useful when the gradient is too laborious or even impossible to derive. This versatility comes at a price, however, as PSO does not always work well and may need tuning of its behavioural parameters so as to perform well on the problem at hand, see for example van den Bergh [3], Trelea [4], Shi and Eberhart [5] [6], Carlisle and Dozier [7], and Clerc [8]. PSO variants are continually being devised in an attempt to overcome this deficiency, see e.g. [9] [10] [11] [12] [13] [14] [15] [16] [17] for a few recent additions. These PSO variants greatly increase the complexity of the original method and we have previously demonstrated that satisfactory performance can be achieved with the basic PSO if only its parameters are properly tuned [18] [19].

This paper gives the practitioner a table of PSO parameters that have been tuned for different optimization scenarios.

2 Particle Swarm Optimization

Consider a fitness (or cost, error, objective) function:

$$f : \mathbb{R}^n \rightarrow \mathbb{R}$$

To minimize the fitness function f find $\vec{a} \in \mathbb{R}^n$ so that:

$$\forall \vec{b} \in \mathbb{R}^n : \ f(\vec{a}) \leq f(\vec{b})$$

Then $\vec{a}$ is called a global minimum for the function f. It is usually not possible to pinpoint the global minimum exactly in optimization and candidate solutions with sufficiently good fitness are deemed acceptable for practical reasons.

In PSO the candidate solutions are the particles and are denoted $\vec{x} \in \mathbb{R}^n$. They are initially placed at random positions in the search-space and moving in randomly defined directions. The direction of a particle is then gradually changed to move in the direction of the best found positions of itself and its peers, searching in their vicinity and potentially discovering better positions.

Small changes to the PSO implementation can cause dramatic changes in the behavioural parameters that cause good optimization performance. The parameters given in this paper have been tuned for the basic PSO algorithm in figure 1. If your PSO implementation differs from this you may need to alter it to use the parameters listed here.

2.1 MOL Variant

A PSO simplification was presented in [18] [19] which eliminates the particle's own best known position $\vec{p}$ by setting $\phi_p = 0$, and in the inner-loop of the algorithm in figure 1 particles are picked randomly. This variant is called Many Optimizing Liaisons (MOL) to make it easy to distinguish from the original PSO. It is slightly easier than PSO to implement and appears to perform just as well, if not better. A similar PSO simplification was suggested by Kennedy [20] who called it the "social only" PSO.

3 Meta-Optimization

The PSO and MOL parameters in tables 1 and 2 have been found by way of meta-optimization, that is, the use of another overlying optimizer to tune the PSO and MOL parameters for different optimization scenarios. The concept is depicted in figure 3 and described in detail in [18].

The PSO and MOL parameters have been tuned for the benchmark problems in table 3 using various dimensionalities and optimization run-lengths. Note that the optimum has been displaced according to the values in table 4 to avoid unintended attraction of the particles to zero which also happens to be the global optimum of most of these benchmark problems. All 12 benchmark problems have been used in meta-optimization to yield behavioural parameters that should work well in general, although for some of the larger meta-optimization scenarios, e.g. the 100 dimensional cases, only the Ackley, Rastrigin, Rosenbrock and Schwefel1-2 problems were used so as to save computation time.

Time usage for meta-optimization of the smallest problem configurations (2 dimensions and 400 fitness evaluations) were mere seconds while up to 24 hours were used for the larger problem configurations when executed on a 1.6

GHz Intel CPU. Using a modern multi-core CPU would decrease the time usage considerably and is readily supported in the source-code linked to below.

4 Example Usage

If you need to optimize a problem using few fitness evaluations, say, a 4-dimensional problem using 100 fitness evalutions, or a 1,000-dimensional problem using 30,000 fitness evaluations, then PSO may not be the right choice of optimizer. Instead you may want to use optimizers that were specifically designed for short optimization runs, see e.g. Pattern Search (PS) and Local Unimodal Sampling (LUS) in [18].

Now assume you are tasked with optimizing a series of problems in 40 dimensions each and you can perform 500,000 fitness evaluations on each problem, what PSO parameters should you use? Consulting table 1 we see that this exact scenario is not listed. The practitioner will then try with the closest match and if that does not yield satisfactory results then try the next closest match, etc. In this case the closest match seems to be the parameters tuned for 30 dimensions and 600,000 fitness evaluations:

$$S = 95,\ \omega = -0.6031,\ \phi_p = -0.6485,\ \phi_g = 2.6475$$

where S is the swarm-size and means there should be 95 particles in the swarm. Similarly the MOL parameters are chosen from table 2 to be:

$$S = 134,\ \omega = -0.4300,\ \phi_g = 3.0469$$

Using these parameters in optimizing the benchmark problems results in table 5 and figures 4 and 5. In this case the results are close to the optimal fitness values of zero which is quite satisfactory (although perhaps not a surprise since the parameters were tuned for these problems), but if the parameters had failed then the practitioner would perhaps try the parameters tuned for 50 dimensions and 100,000 fitness evaluations, or 20 dimensions and 400,000 fitness evaluations. If those failed as well then the practitioner would need to either meta-optimize the PSO (or MOL) parameters for the problem at hand or use another optimization method.

5 Conclusion

This paper presented a table of PSO parameters that may be used by the practitioner as a first choice when optimizing new problems. The parameters were tuned (meta-optimized) to perform well on several benchmark problems with various dimensionalities and optimization run-lengths.

6 Source-Code

Source-code implemented in the C# programming language and used in the experiments in this paper can be found in the SwarmOps library on the internet address: `http://www.Hvass-Labs.org/`

References

[1] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume IV, pages 1942–1948, Perth, Australia, 1995.

[2] Y. Shi and R.C. Eberhart. A modified particle swarm optimizer. In *Proceedings of 1998 IEEE International Conference on Evolutionary Computation*, pages 69–73, Anchorage, AK, USA, 1998.

[3] F. van den Bergh. *An Analysis of Particle Swarm Optimizers*. PhD thesis, University of Pretoria, Faculty of Natural and Agricultural Science, November 2001.

[4] I.C. Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters*, 85:317 – 325, 2003.

[5] Y. Shi and R.C. Eberhart. Parameter selection in particle swarm optimization. In *Proceedings of Evolutionary Programming VII (EP98)*, pages 591 – 600, 1998.

[6] R.C. Eberhart and Y. Shi. Comparing inertia weights and constriction factors in particle swarm optimization. *Proceedings of the 2000 Congress on Evolutionary Computation*, 1:84 – 88, 2000.

[7] A. Carlisle and G. Dozier. An off-the-shelf PSO. In *Proceedings of the Particle Swarm Optimization Workshop*, pages 1 – 6, 2001.

[8] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6:58 – 73, 2002.

[9] Z-H. Zhan, J. Zhang, Y. Li, and H.S-H. Chung. Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, 39:1362–1381, 2009.

[10] Z. Xinchao. A perturbed particle swarm algorithm for numerical optimization. *Applied Soft Computing*, 10:119–124, 2010.

[11] T. Niknam and B. Amiri. An efficient hybrid approach based on PSO, ACO and k-means for cluster analysis. *Applied Soft Computing*, 10:183–197, 2010.

[12] M. El-Abda, H. Hassan, M. Anisa, M.S. Kamela, and M. Elmasry. Discrete cooperative particle swarm optimization for FPGA placement. *Applied Soft Computing*, 10:284–295, 2010.

[13] M-R. Chena, X. Lia, X. Zhanga, and Y-Z. Lu. A novel particle swarm optimizer hybridized with extremal optimization. *Applied Soft Computing*, 10:367–373, 2010.

[14] P.W.M. Tsang, T.Y.F. Yuena, and W.C. Situ. Enhanced affine invariant matching of broken boundaries based on particle swarm optimization and the dynamic migrant principle. *Applied Soft Computing*, 10:432–438, 2010.

[15] C-C. Hsua, W-Y. Shiehb, and C-H. Gao. Digital redesign of uncertain interval systems based on extremal gain/phase margins via a hybrid particle swarm optimizer. *Applied Soft Computing*, 10:606–612, 2010.

[16] H. Liua, Z. Caia, and Y. Wang. Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Applied Soft Computing*, 10:629–640, 2010.

[17] K. Mahadevana and P.S. Kannan. Comprehensive learning particle swarm optimization for reactive power dispatch. *Applied Soft Computing*, 10:641–652, 2010.

[18] M.E.H. Pedersen. *Tuning & Simplifying Heuristical Optimization.* PhD thesis, School of Engineering Sciences, University of Southampton, England, 2010.

[19] M.E.H. Pedersen and A.J. Chipperfield. Simplifying particle swarm optimization. *Applied Soft Computing*, 10:618–628, 2010.

[20] J. Kennedy. The particle swarm: social adaptation of knowledge. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 303–308, Indianapolis, USA, 1997.

- Initialize each particle $\vec{x} \in \mathbb{R}^n$ with a random position in the search-space:

$$\vec{x} \sim U(\vec{b}_{lo}, \vec{b}_{up})$$

  where $\vec{b}_{lo}$ and $\vec{b}_{up}$ are the lower and upper boundaries of the search-space.
- Set each particle's best known position to its initial position:

$$\vec{p} \leftarrow \vec{x}$$

- Initialize each particle's velocity $\vec{v} \in \mathbb{R}^n$ to random values:

$$\vec{v} \sim U(-\vec{d}, \vec{d})$$

  where $\vec{d} = |\vec{b}_{up} - \vec{b}_{lo}|$
- Initialize the swarm's best known position $\vec{g}$ to the $\vec{x}$ for which $f(\vec{x})$ is lowest.
- Until a termination criterion is met, repeat the following:
  - For each particle $\vec{x}$ in the swarm do the following:
    - Pick two random numbers: r_p, $r_g \sim U(0, 1)$
    - Update the particle's velocity $\vec{v}$ as follows:

$$\vec{v} \leftarrow \omega\vec{v} + \phi_p r_p(\vec{p} - \vec{x}) + \phi_g r_g(\vec{g} - \vec{x})$$

      where ω, ϕ_p, and ϕ_g are user-defined behavioural parameters.
    - Bound the velocity, that is, for all dimensions i update v_i:

$$v_i \leftarrow \text{Bound}(v_i, -d_i, d_i)$$

      See figure 2 for the definition of Bound()
    - Move the particle to its new position by adding its velocity:

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

    - Bound the position, that is, for all dimensions i update x_i:

$$x_i \leftarrow \text{Bound}(x_i, b_{lo_i}, b_{up_i})$$

    - If $(f(\vec{x}) < f(\vec{p}))$ then update the particle's best known position:

$$\vec{p} \leftarrow \vec{x}$$

    - If $(f(\vec{x}) < f(\vec{g}))$ then update the swarm's best known position:

$$\vec{g} \leftarrow \vec{x}$$

- Now $\vec{g}$ holds the best found position in the search-space.

Figure 1: PSO pseudo-code.

Problem Dimensions	Fitness Evaluations	PSO Parameters			
		S	ω	ϕ_p	ϕ_g
2	400	25	0.3925	2.5586	1.3358
		29	-0.4349	-0.6504	2.2073
2	4,000	156	0.4091	2.1304	1.0575
		237	-0.2887	0.4862	2.5067
5	1,000	63	-0.3593	-0.7238	2.0289
		47	-0.1832	0.5287	3.1913
5	10,000	223	-0.3699	-0.1207	3.3657
		203	0.5069	2.5524	1.0056
10	2,000	63	0.6571	1.6319	0.6239
		204	-0.2134	-0.3344	2.3259
10	20,000	53	-0.3488	-0.2746	4.8976
20	40,000	69	-0.4438	-0.2699	3.3950
20	400,000	149	-0.3236	-0.1136	3.9789
		60	-0.4736	-0.9700	3.7904
		256	-0.3499	-0.0513	4.9087
30	600,000	95	-0.6031	-0.6485	2.6475
50	100,000	106	-0.2256	-0.1564	3.8876
100	200,000	161	-0.2089	-0.0787	3.7637

Table 1: PSO parameters for various problem configurations. The practitioner should select the PSO parameters where the dimensionality and allowed number of fitness evaluations most closely match those of the optimization problem at hand. For some problem configurations multiple parameters are listed as they had almost the same optimization performance.

Problem Dimensions	Fitness Evaluations	MOL Parameters		
		S	ω	ϕ_g
2	400	23	-0.3328	2.8446
		50	0.2840	1.9466
2	4,000	183	-0.2797	3.0539
		139	0.6372	1.0949
5	1,000	50	-0.3085	2.0273
5	10,000	96	-0.3675	4.171
10	2,000	60	-0.2700	2.9708
10	20,000	116	-0.3518	3.8304
20	40,000	228	-0.3747	4.2373
20	400,000	125	-0.2575	4.6713
		67	-0.4882	2.7923
30	600,000	134	-0.4300	3.0469
50	100,000	290	-0.3067	3.6223
100	200,000	219	-0.1685	3.9162

Table 2: MOL parameters for various problem configurations. The practitioner should select the MOL parameters where the dimensionality and allowed number of fitness evaluations most closely match those of the optimization problem at hand. For some problem configurations multiple parameters are listed as they had almost the same optimization performance.

$$\text{Bound}(x, l, u) = \begin{cases} l & , x < l \\ u & , x > u \\ x & , else \end{cases}$$

Figure 2: Bounding function used in the PSO algorithm.

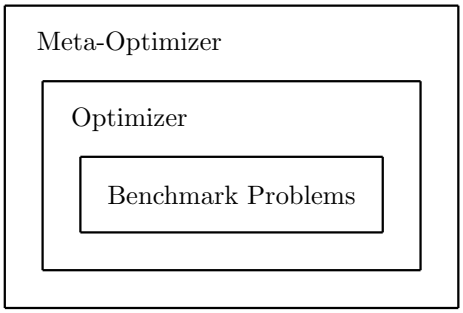

Figure 3: The concept of meta-optimization. Another optimization method is used as an overlying meta-optimizer for finding good behavioural parameters of PSO (or MOL), which in turn is used to optimize benchmark problems.

Ackley	$f(\vec{x}) = e + 20 - 20 \cdot \exp\left(-0.2 \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)\right)$
Griewank	$f(\vec{x}) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)$
Penalized1	$f(\vec{x}) = \frac{\pi}{n}\left(10 \cdot \sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2 \cdot \left(1 + 10 \cdot \sin^2(\pi y_{i+1})\right) + (y_n - 1)^2\right) + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + (x_i + 1)/4$ $u(x_i, a, k, m) = \begin{cases} k(-x_i - a)^m & , x_i < -a \\ 0 & , -a \le x_i \le a \\ k(x_i - a)^m & , x_i > a \end{cases}$
Penalized2	$f(\vec{x}) = 0.1\left(\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2 \cdot \left(1 + \sin^2(3\pi x_{i+1})\right) + (x_n - 1)^2 \cdot \left(1 + \sin^2(2\pi x_n)\right)\right) + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$, with $u(\cdot)$ from above.
QuarticNoise	$f(\vec{x}) = \sum_{i=1}^{n}(i \cdot x_i^4 + r_i),\ r_i \sim U(0, 1)$
Rastrigin	$f(\vec{x}) = \sum_{i=1}^{n}\left(x_i^2 + 10 - 10 \cdot \cos(2\pi x_i)\right)$
Rosenbrock	$f(\vec{x}) = \sum_{i=1}^{n-1}\left(100 \cdot (x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)$
Schwefel1-2	$f(\vec{x}) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$
Schwefel2-21	$f(\vec{x}) = \max\{\lvert x_i\rvert : i \in \{1, \cdots, n\}\}$
Schwefel2-22	$f(\vec{x}) = \sum_{i=1}^{n}\lvert x_i\rvert + \prod_{i=1}^{n}\lvert x_i\rvert$
Sphere	$f(\vec{x}) = \sum_{i=1}^{n} x_i^2$
Step	$f(\vec{x}) = \sum_{i=1}^{n}\left(\lfloor x_i + 0.5\rfloor\right)^2$

Table 3: Benchmark problems.

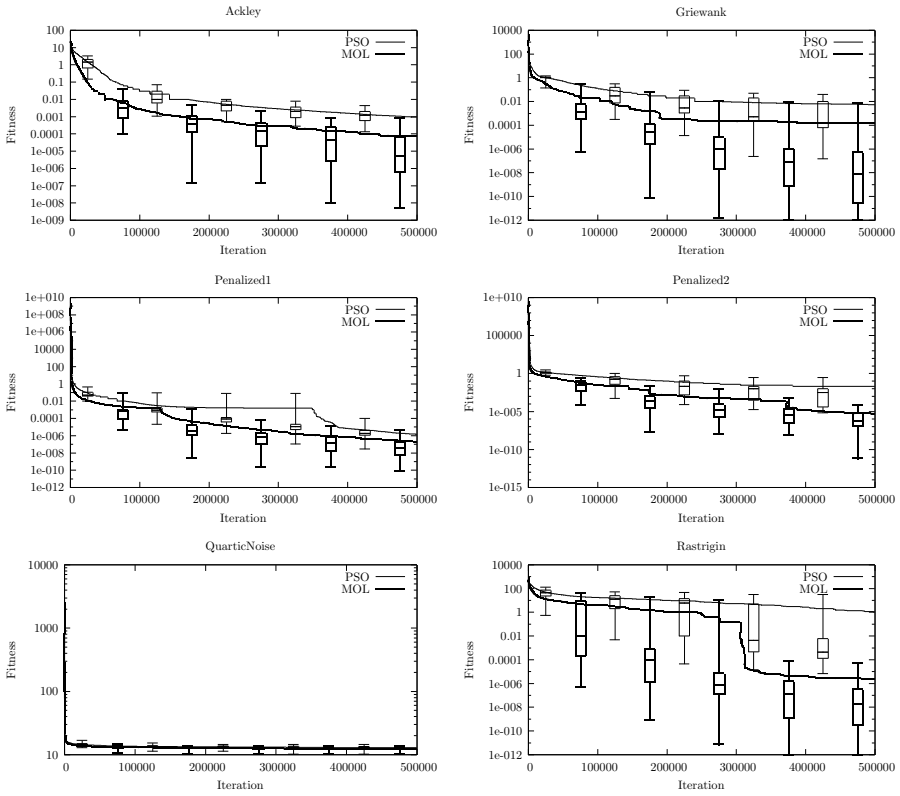

Figure 4: PSO and MOL optimization performance. Plots show the mean fitness achieved over 50 optimization runs as well as the quartiles at intervals during optimization.

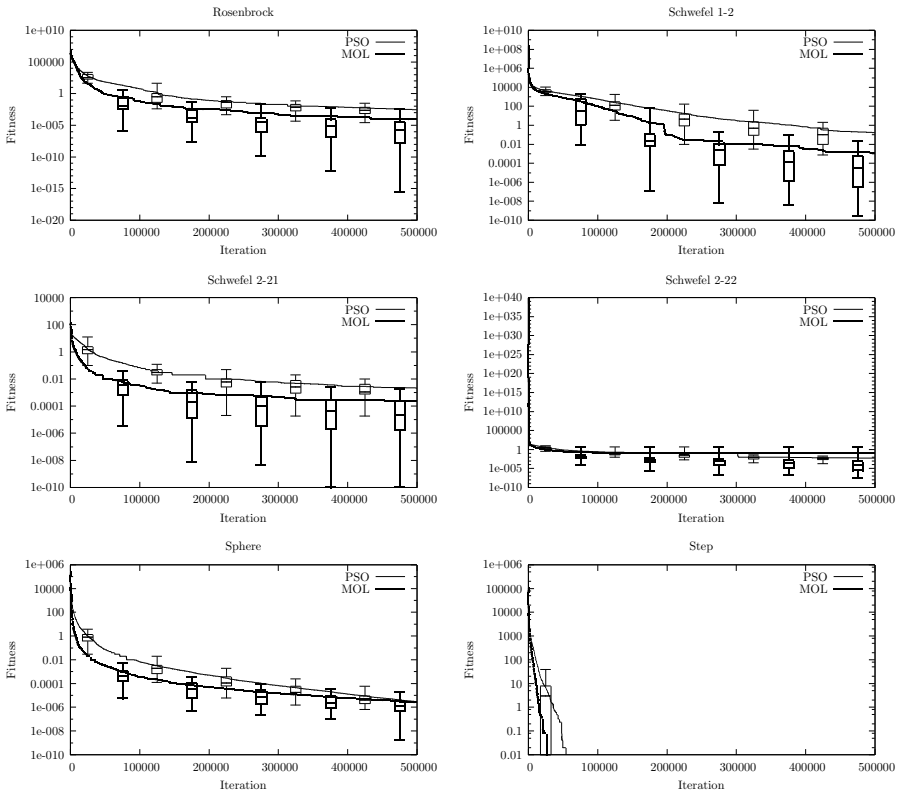

Figure 5: PSO and MOL optimization performance. Plots show the mean fitness achieved over 50 optimization runs as well as the quartiles at intervals during optimization.

Problem	Initialization	Search-Space	Displacement δ
Ackley	$[15, 30]$	$[-30, 30]$	-7.5
Griewank	$[300, 600]$	$[-600, 600]$	-150
Penalized1	$[5, 50]$	$[-50, 50]$	0
Penalized2	$[5, 50]$	$[-50, 50]$	0
QuarticNoise	$[0.64, 1.28]$	$[-1.28, 1.28]$	-0.32
Rastrigin	$[2.56, 5.12]$	$[-5.12, 5.12]$	1.28
Rosenbrock	$[15, 30]$	$[-100, 100]$	25
Schwefel1-2	$[50, 100]$	$[-100, 100]$	-25
Schwefel2-21	$[50, 100]$	$[-100, 100]$	-25
Schwefel2-22	$[5, 10]$	$[-10, 10]$	-2.5
Sphere	$[50, 100]$	$[-100, 100]$	25
Step	$[50, 100]$	$[-100, 100]$	25

Table 4: Initialization ranges, search-space boundaries, and displacement values δ for the benchmark problems. Displacement is done by using an auxiliary fitness function $h(\vec{x}) = f(\vec{x} - \delta)$ to avoid unintended attraction of PSO and MOL particles to the zero-position which happens to be the optimal solution for most of these problems.

	Problem	Mean	Std.Dev.	Min	Q1	Median	Q3	Max
PSO	Ackley	9.36e-4	6.86e-4	3.98e-5	4.31e-4	8.1e-4	1.37e-3	3.41e-3
	Griewank	5.4e-3	9.81e-3	1.15e-7	2.37e-5	7.03e-5	8.58e-3	0.04
	Penalized1	1.48e-6	3.61e-6	1.18e-8	3.3e-7	7.91e-7	1.59e-6	2.6e-5
	Penalized2	0.02	0.05	2.73e-6	1.41e-5	1.98e-4	0.01	0.3
	QuarticNoise	12.93	0.79	10.19	12.53	12.94	13.55	14.52
	Rastrigin	1.1	4.31	2.85e-6	2.71e-5	1.51e-4	9.26e-4	27.87
	Rosenbrock	2.9e-3	3.89e-3	2.11e-5	3.69e-4	1.22e-3	3.52e-3	0.02
	Schwefel1-2	0.17	0.26	5.67e-4	5.53e-3	0.05	0.25	1.2
	Schwefel2-21	2.15e-3	2.85e-3	1.45e-5	4.14e-4	8.85e-4	2.29e-3	0.01
	Schwefel2-22	5.55e-3	5.09e-3	1.85e-4	1.21e-3	3.79e-3	8.71e-3	0.02
	Sphere	2.99e-6	4.5e-6	2.13e-7	9.32e-7	1.48e-6	3.12e-6	2.67e-5
	Step	0	0	0	0	0	0	0
MOL	Ackley	7.32e-5	1.5e-4	5.45e-9	5.69e-7	5.42e-6	6.93e-5	8.6e-4
	Griewank	1.51e-4	1.06e-3	0	2.5e-11	6.68e-9	4.16e-7	7.55e-3
	Penalized1	2.05e-7	6.62e-7	7.17e-11	3.67e-9	2.59e-8	1.25e-7	4.63e-6
	Penalized2	5.3e-6	1.07e-5	6.43e-12	1.06e-7	5.24e-7	5.57e-6	5.7e-5
	QuarticNoise	12.45	0.63	10.31	12.09	12.56	12.85	13.74
	Rastrigin	2.41e-6	9.74e-6	0	9.64e-11	7.18e-9	2.81e-7	5.53e-5
	Rosenbrock	8.89e-5	4.84e-4	3.27e-16	3.42e-9	6.87e-7	3.16e-5	3.47e-3
	Schwefel1-2	1.21e-3	3.38e-3	2.63e-10	2.58e-7	3e-5	5.78e-4	0.02
	Schwefel2-21	2.37e-4	4.34e-4	1.02e-10	8.08e-7	2.04e-5	2.36e-4	1.77e-3
	Schwefel2-22	0.1	0.7	3.69e-8	3e-6	9.07e-5	7.39e-4	5
	Sphere	2.6e-6	3.76e-6	1.59e-9	4.38e-7	1.01e-6	2.85e-6	1.82e-5
	Step	0	0	0	0	0	0	0

Table 5: Optimization end results for PSO and MOL when the benchmark problems have 40 dimensions and 500,000 fitness evaluations have been performed.