A Comparison of Particle Swarm Optimization Algorithms Based on Run-Length Distributions

Marco A. Montes de Oca, Thomas Stützle, Mauro Birattari, and Marco Dorigo

IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
{mmontes,stuetzle,mbiro,mdorigo}@ulb.ac.be

Abstract. In this paper we report an empirical comparison of some of the most influential Particle Swarm Optimization (PSO) algorithms based on run-length distributions (RLDs). The advantage of our approach over the usual report pattern (average iterations to reach a predefined goal, success rates, and standard deviations) found in the current PSO literature is that it is possible to evaluate the performance of an algorithm on different application scenarios at the same time. The RLDs reported in this paper show some of the strengths and weaknesses of the studied algorithms and suggest ways of improving their performance.

1 Introduction

Since the introduction of the first Particle Swarm Optimization (PSO) algorithm by Kennedy and Eberhart [1, 2], many variants of the original algorithm have been proposed. The approach followed by many researchers to evaluate the performance of their variants has been to compare the proposed variant with the original version or, more recently, with the so-called canonical version [3]. In many cases, these new variants are reported to perform better, see for instance [4–7].

Unfortunately, since there are no cross-comparisons among variants, there is no general agreement on which PSO variant(s) could be considered the state-of-the-art in the field. The motivation for conducting the comparison reported in this paper is the identification of these variant(s). However, determining the state-of the-art algorithm is not a trivial task. In particular, one must be aware of the possible application scenarios in which a stochastic optimization algorithm may be used. Our main concern can be solution quality, time or both. Of course, in any case, the sooner we get a solution, the better. However, because of the stochastic nature of these algorithms, finding a high quality solution in a timely fashion only happens with a certain probability. Characterizing the distribution of this probability is the purpose of the run-time distribution. Formally, a stochastic optimization algorithm A applied to a problem Π will find a solution of quality q in time t with probability $P_{A,\Pi}(q,t) = P(RT_{A,\Pi} \leq t, SQ_{A,\Pi} \leq q)$, and the bivariate random variable $(RT_{A,\Pi}, SQ_{A,\Pi})$ describes the run-time and solution quality behavior of an algorithm A when applied to problem Π; the probability distribution of this random variable is also known as the run-time

distribution of A on Π [8]. Since in continuous optimization we measure run-time in terms of the number of function evaluations, we talk of *run-length* distributions (RLDs) rather than *run-time* distributions. This is the approach followed in this paper.

An RLD completely characterizes the performance of a stochastic optimization algorithm on a particular problem, regardless of the actual application scenario in which we may be interested in. We say this because with an RLD we can estimate the probability of finding a solution of a certain quality given some time limit. This is the main reason why we chose to evaluate some of the most influential PSO algorithms using RLDs. As a bonus, an analysis based on RLDs allows the identification of some strengths and weaknesses of the studied algorithms and may also be used to design improved versions.

The rest of the paper is organized as follows. Section 2 briefly describes the PSO technique and the variants compared in this paper. Section 3 describes the experimental setup adopted for our comparison. Section 4 presents the development of the solution quality over time and the RLDs of the studied algorithms. Section 5 summarizes the main contributions and results presented in the paper.

2 Particle Swarm Optimization Algorithms

In the original PSO algorithm [1, 2], a fixed number of solutions (called *particles* in a PSO context) are randomly initialized in a d-dimensional solution space. A particle i at time step t has a position vector $\boldsymbol{x}_i^t$ and a velocity vector $\boldsymbol{v}_i^t$. An objective function $f : S \rightarrow \Re$, with $S \subset \Re^d$, determines the quality of a particle's position, i.e., a particle's position represents a solution to the problem being solved. Each particle i has a vector $\boldsymbol{p}_i$ that represents its own best previous position that has an associated objective function value $pbest_i = f(\boldsymbol{p}_i)$. Finally, the best position the swarm has ever visited is stored in a vector $\boldsymbol{s}$ whose objective function value is $gbest = f(\boldsymbol{s})$.

The algorithm iterates updating the velocities and positions of the particles until a stopping criterion is met. The update rules are:

$$\boldsymbol{v}_i^{t+1} = \boldsymbol{v}_i^t + \varphi_1 \boldsymbol{U}_1(0,1) * (\boldsymbol{p}_i - \boldsymbol{x}_i^t) + \varphi_2 \boldsymbol{U}_2(0,1) * (\boldsymbol{s} - \boldsymbol{x}_i^t)\,, \tag{1}$$

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1}\,, \tag{2}$$

where φ_1 and φ_2 are two constants called the *cognitive* and *social* acceleration coefficients respectively, $\boldsymbol{U}_1(0,1)$ and $\boldsymbol{U}_2(0,1)$ are two d-dimensional uniformly distributed random vectors in which each component goes from zero to one, and $*$ is an element-by-element vector multiplication operator.

The variants we compare in this study were selected either because they are among the most commonly used in the field or because they look very promising. In the following subsections, we describe them in more detail.

2.1 Canonical Particle Swarm Optimizer

Clerc and Kennedy [3] introduced a constriction factor into PSO to control the convergence properties of the particles. This constriction factor is added in Equation 1 giving

$$\boldsymbol{v}_i^{t+1} = \chi \left(\boldsymbol{v}_i^t + \varphi_1 \boldsymbol{U}_1(0,1) * (\boldsymbol{p}_i - \boldsymbol{x}_i^t) + \varphi_2 \boldsymbol{U}_2(0,1) * (\boldsymbol{s} - \boldsymbol{x}_i^t) \right) , \qquad (3)$$

with

$$\chi = 2k / \left(\left| 2 - \varphi - \sqrt{\varphi^2 - 4\varphi} \right| \right) , \qquad (4)$$

where $k \in [0,1]$, $\varphi = \varphi_1 + \varphi_2$ and $\varphi > 4$. Usually, k is set to 1 and both φ_1 and φ_2 are set to 2.05, giving as a result χ equal to 0.729 [9, 10]. This variant has been so widely used that it is known as the *canonical* PSO.

2.2 Time-Varying Inertia Weight Particle Swarm Optimizer

Shi and Eberhart [4, 11] introduced the idea of a time-varying inertia weight. The idea was to control the diversification–intensification behavior of the original PSO. The velocity update rule is

$$\boldsymbol{v}_i^{t+1} = w(t)\boldsymbol{v}_i^t + \varphi_1 \boldsymbol{U}_1(0,1) * (\boldsymbol{p}_i - \boldsymbol{x}_i^t) + \varphi_2 \boldsymbol{U}_2(0,1) * (\boldsymbol{s} - \boldsymbol{x}_i^t) , \qquad (5)$$

where $w(t)$ is the time-varying inertia weight which usually is linearly adapted from an initial value to a final one. In most cases, φ_1 and φ_2 are both set to 2.

There are two ways of varying the inertia weight in time: decreasingly (e.g., as in [4, 12, 11]) and increasingly (e.g., as in [13, 14]). In this paper, we included both variants for the sake of completeness. Normally, the starting value of the inertia weight is set to 0.9 and the final to 0.4. Zheng et al. [13, 14], use the opposite settings. In the results section, these variants are identified by Dec-IW and Inc-IW, respectively.

2.3 Stochastic Inertia Weight Particle Swarm Optimizer

Eberhart and Shi [15] proposed another variant in which the inertia weight is randomly selected according to a uniform distribution in the range [0.5,1.0]. This range was inspired by Clerc and Kennedy's constriction factor. In this version, the acceleration coefficients are set to 1.494 as a result of the multiplication $\chi \cdot \varphi_{1,2}$. Although this variant was originally proposed for dynamic environments, it has also been shown to be a competitive optimizer for static ones [16]. In the results section this variant is identified by Sto-IW.

2.4 Fully Informed Particle Swarm Optimizer

In the fully informed particle swarm (FIPS) proposed by Mendes et al. [7], a particle uses information from all its topological neighbors. This variant is based

on the fact that Clerc and Kennedy's constriction factor does not enforce that the value φ should be split only between two attractors.

For a given particle, the way φ (i.e., the sum of the acceleration coefficients) is decomposed is $\varphi_k = \varphi/|\mathcal{N}| \; \forall k \in \mathcal{N}$ where $\mathcal{N}$ is the neighborhood of the particle. As a result, the new velocity update equation becomes

$$\boldsymbol{v}_i^{t+1} = \chi \left[\boldsymbol{v}_i^t + \sum_{k \in \mathcal{N}} \varphi_k \mathcal{W}(k) \boldsymbol{U}_k(0,1) * (\boldsymbol{p}_k - \boldsymbol{x}_i^t) \right], \tag{6}$$

where $\mathcal{W}(k)$ is a weighting function.

2.5 Self-Organizing Hierarchical Particle Swarm Optimizer with Time-varying Acceleration Coefficients

The self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients (HPSOTVAC) proposed by Ratnaweera et al. [16] drops the velocity term from the right side of Equation 5. If a particle's new velocity becomes zero (in any dimension), it is reinitialized to some value proportional to the maximum allowable velocity V_{max}. HPSOTVAC linearly adapts the value of the acceleration coefficients φ_1 and φ_2 to enforce the diversification behavior at the beginning of the run and the intensification behavior at the end. φ_1 is decreased from 2.5 to 0.5 and φ_2 increases from 0.5 to 2.5. Finally, the reinitialization velocity is also linearly decreased from V_{max} at the beginning of the run to $0.1 \cdot V_{max}$ at the end.

2.6 Adaptive Hierarchical Particle Swarm Optimizer

Proposed by Janson and Middendorf [17], the adaptive hierarchical PSO (AHPSO) is an example of a PSO with dynamic adaptation of the population topology. In AHPSO, the topology is a tree-like structure in which particles with a higher fitness evaluation are located in the upper nodes of the tree. At each iteration, a child particle updates its velocity considering its own previous best performance and the previous best performance of its parent. Additionally, before the velocity updating process takes place, the previous best fitness value of any particle is compared with that of its parent. If it is better, child and parent swap their positions in the hierarchy.

The branching degree of the tree is a factor that can balance the diversification-intensification behavior of the algorithm. To dynamically adapt the algorithm to the stage of the optimization process, the branching degree is decreased by k_{adapt} degrees until a certain minimum degree d_{min} is reached. This process takes place every f_{adapt} number of iterations. The parameters that control this process need to be tuned for each problem [17]. In our experiments, for the reasons explained in the next section, we set the initial branching factor to 20, parameters d_{min}, f_{adapt}, and k_{adapt} were set to 2, $1000*m$, and 3 respectively, where m is the number of particles.

3 Experimental Setup

All the PSO variants described in the previous section were implemented for this comparison. To ensure the correctness of our implementations, we tested them on the same problems with the same parameters as reported in the literature[1]. To allow the comparison of the results with previous works, we used some of the most common benchmark functions in the PSO literature: Sphere, Rosenbrock, Rastrigin, Griewank, and Schaffer's F6 functions in 30 dimensions. The mathematical definition of these functions is readily available in the literature (cf. [10]). In our runs, these functions were shifted and biased exactly as specified in [18][2]. Because of this, our initializations are, in all cases, asymmetric with respect to the global optimum.

The reported results are based on 100 independent trials of 1 000 000 function evaluations. In our experiments, we used swarms of 20 particles using two different topologies: fully connected and ring with unitary neighborhood size. The results are organized by population topology. Both topologies included self-references (i.e., every particle is a neighbor to itself). This separation was needed to highlight the influence of the used topology in the behavior of the algorithms. Note that the AHPSO algorithm uses neither a fully connected topology nor a ring topology and therefore appears in both sets of results.

Before proceeding to the presentation of our results, it is worth noting that most PSO algorithms are not robust in their parameterization. For example, in the PSO variants based on a time-varying inertia weight, the slope of the increasing or decreasing inertia weight function is determined by the maximum number of function evaluations. Another problem (for comparison purposes) is that it is also possible to fine-tune the parameters of a variant to solve a particular problem. A possible solution to this problem is to fine-tune all variants for the problem at hand and proceed with the comparison; however, if our aim is to solve real-world problems which generally have a structure we do not know in advance, we need algorithms with a set of "normally good" parameters. For this reason, in this study each algorithm used the same parameterization across the benchmark problems. The actual values chosen for the parameters have already been mentioned in the preceding sections.

4 Results

Tables 1 and 2 show the average value and standard deviation of the number of function evaluations needed to achieve a certain solution quality with fully connected and ring topologies, respectively. For each function, there are three different solution qualities. The first one corresponds to the usual goal for that function (cf. [10]). The second and third can be considered medium and high solution qualities, respectively. The absolute values can be computed as follows:

[1] For space restrictions, we refer the interested reader to the following address: http://iridia.ulb.ac.be/supp/IridiaSupp2006-003/index.html

[2] The values of the optima are specified in Tables 1 and 2.

if, for example, the desired solution quality is 0.01% and the optimum is at -130.0, it means that the goal to reach is $-130 - (0.0001 \times -130.0) = -129.987$.

Table 1. Average value and standard deviation of the number of function evaluations needed to achieve a certain solution quality (S.Q.) using the fully connected topology. Only successful runs are considered. {f1=Griewank (optimum at -130.0), f2=Rastrigin (optimum at -330), f3=Rosenbrock (optimum at 390), f4=Schaffer's F6 (optimum at -300), f5=Sphere (optimum at -450)}

Function	S.Q. (%)	AHPSO	Canonical	FIPS	Dec-IW	HPSOTVAC	Inc-IW	Sto-IW
f1	0.077	9641.3	**8345.8**	–	433036	25741.8	8365.5	9713.8
		1413	1271.4	–	24012.3	1647.1	852.8	1483.5
	0.01	11613.8	**9784.1**	–	442995	34474.3	10115.8	11806.7
		1508.1	1153	–	15996.2	7430.5	1274.6	1789
	0.001	12994.3	**11322.7**	–	453478	57830	41082.6	13306.2
		1481	2120.2	–	21515.5	61113.8	166025	2335.8
f2	30.30	4011	3836.6	**960**	364350	29852.8	53104.4	4820.7
		1478.6	1065.2	56.5	42590.1	21636.7	212169	1534.7
	15	6295	**5550**	–	427895	101322	516798	7758
		1123	1400.1	–	30835.2	44487.3	490241	2554.6
	1	–	–	–	–	**635060**	–	–
		–	–	–	–	133964	–	–
f3	25.64	108546	50148.6	–	492989	489920	**22381.2**	56747.8
		150900	58874	–	70214.1	258767	19505.5	105101
	10	153596	87750.5	–	548089	665170	**67398.8**	95153.8
		204835	90431.8	–	85071.9	205934	104077	148441
	1	353182	**168799**	–	653042	762488	288477	232858
		201800	97989.6	–	116809	218223	208034	152029
f4	3.3×10^{-6}	33636.3	21044.6	**5230**	115876	84893.3	72287.1	56837.6
		93014.2	44560.4	3921.3	24171.4	184640	196370	172404
	1×10^{-7}	34306.1	21478.2	**5540**	131253	86826	58508.4	57350.8
		93024.3	44536.7	3964.9	22963.3	184097	161070	172359
	1×10^{-8}	34540	21789.6	**5724**	140434	88095.1	58760.6	57698.4
		92995.1	44506.6	3932.1	19112.8	183701	161142	172325
f5	0.0022	11342.8	10913	–	433345	31210.4	**8135.1**	11127.8
		1386.8	2255.7	–	6885.3	1436.4	944.4	1317
	0.0002	14000.6	13122.4	–	446824	40913	**9772**	13680.4
		1468.2	2648	–	6707.7	1616.7	946	1919
	0.00001	15862.6	14955.8	–	456062	48546.6	**11147.4**	15469.6
		1506.1	2795.7	–	6346.5	1723	1152	1890.7

Most variants, most notably FIPS, are greatly affected in their performance by the used topology. With a fully connected topology, most of the tested variants reach the specified solution quality faster than with the ring topology. FIPS performs poorly with this topology: only in 4 out of 15 cases it reaches the specified solution quality. However, whenever it does, it is the fastest algorithm.

Table 2. Average value and standard deviation of the number of function evaluations needed to achieve a certain solution quality (S.Q.) using the ring topology. Only successful runs are considered. {f1=Griewank (optimum at -130.0), f2=Rastrigin (optimum at -330), f3=Rosenbrock (optimum at 390), f4=Schaffer's F6 (optimum at -300), f5=Sphere (optimum at -450)}

Function	S.Q. (%)	AHPSO	Canonical	FIPS	Dec-IW	HPSOTVAC	Inc-IW	Sto-IW
f1	0.077	9641.3	12377.8	**8030.2**	456568	29862	17268.8	15530.2
		1413	849	957.8	11431.9	1538.6	1192.3	1538.2
	0.01	**11613.8**	15865.6	12454.5	499384	40342.2	29852.1	23091.9
		1508.1	4322.2	7074.7	68132.2	3660.7	50030.8	37164.4
	0.001	**12994.3**	32157.9	21422.5	536619	60952.1	61165.4	35051.8
		1481	62991	26631.3	90978.1	36481	115035	71418.5
f2	30.30	**4011**	30091.3	22599.6	360126	30052.8	34257.8	19767.8
		1478.6	89696.1	9998.2	63803.9	12946.6	142877	84546.4
	15	**6295**	–	136648	460690	117057	685132	206669
		1123	–	100821	59069.9	37364.8	429521	275732
	1	–	–	–	–	**811247**	–	–
		–	–	–	–	107569	–	–
f3	25.64	108546	**104684**	226828	518732	361997	127764	151060
		150900	130114	252722	70783.1	325196	154960	188680
	10	**153596**	189872	320153	605140	291822	173475	215458
		204835	216622	270742	124911	263481	166956	233277
	1	**353182**	426285	443895	734466	–	531935	458950
		201800	221124	242211	146611	–	270172	262995
f4	3.3×10^{-6}	33636.3	43691.4	40416.9	123649	126988	**28014**	40241.2
		93014.2	89702.5	90967.5	28220.1	204489	32195.1	79386.6
	1×10^{-7}	34306.1	44897.6	49353.5	139871	129022	**29321.8**	42340.2
		93024.3	89965.7	92344.4	27231.6	204003	32216.9	82079
	1×10^{-8}	34540	45482.8	54939.6	150797	130576	**29771.6**	43142.4
		92995.1	89856.7	92882.9	27877.7	203686	32327.5	82122.1
f5	0.0022	11342.8	13693.6	**8266**	459971	35518.2	14923	17148.6
		1386.8	572.3	480.3	9304.4	1382.3	894.8	1129.21
	0.0001	14000.6	16421.4	**9920.8**	477006	47480	18077.4	20660.8
		1468.2	663.3	491.7	8254.2	1469.7	956.3	1265.5
	0.00001	15862.6	18484.2	**11169.6**	488408	56889	20430.6	23289.4
		1506.1	684.3	523.2	7977.2	1923.1	1055	1344.9

The data shown in Tables 1 and 2 should be taken *cum grano salis*. The averages and standard deviations reported there are computed over successful runs only. Since these data alone can be misleading, the median solution quality over time (not included in the paper due to space restrictions) is reported in the already mentioned URL. However, the good performance of FIPS using the ring topology is confirmed by the median. FIPS is among the fastest variants.

HPSOTVAC is the only variant that is able to find the highest solution quality target in the Rastrigin function. HPSOTVAC succeeds at reaching the goal but spends many function evaluations to do that.

AHPSO performs relatively better than the other variants when they use the ring topology. This is expected since AHPSO adapts the population hierarchy from a highly connected one to a loosely connected one, so it exploits the benefits of converging faster at the beginning of the run. As seen from the results, fast convergence is somehow associated to the fully connected topology, or at least with a highly connected one.

The Canonical PSO and the Increasing Inertia Weight variants perform pretty well. With the fully connected topology, they are the best performers in 10 out of 15 cases. With the ring topology, this number drops to only 4. These variants clearly exploit the convergence properties of the fully connected topology.

In this paper we report *qualified* RLDs which are cross-sections along the computing time axis of the full joint distribution of the bivariate random variable $(RT_{A,\Pi}, SQ_{A,\Pi})$ described in Section 1. The interested reader is referred to [8] for more information about RLDs. Figures 1 and 2 show the RLDs in four benchmark functions. The shown RLDs correspond to solution qualities of 0.01% for Griewank function, 30% for Rastrigin function, 10% for Rosenbrock function, and 0.0001% for Schaffer's F6 function. The results are organized by population topology: on the left, the results obtained using a fully connected topology; on the right, using the ring topology.

The "slope" of the shown curves point out interesting features of the algorithms. If an RLD for a given solution quality is steep (but complete), it means that the algorithm finds the solution easily. If the demanded solution quality is high, the algorithm will need more function evaluations to find it. This will cause the curve to change its position (the higher the quality, the more to the right) and, possibly, its slope. This is the case with the HPSOTVAC variant using the ring topology in the Griewank function (Figure 1,(b)). It can be seen, however, that this is an exception and not the rule. Most variants have curves with low steepness or steep incomplete curves which is an indication that in some trials the algorithm gets stuck in some local optima far from the target solution quality.

An analysis based on RLDs allows us to measure the severity of search stagnation experienced by optimization algorithms and lets us devise ways to counteract it. For example, all variants suffer from severe stagnation when solving Griewank and Rastrigin problems. To counteract it, they could use a restarting mechanism as suggested by Hoos and Stützle [8].

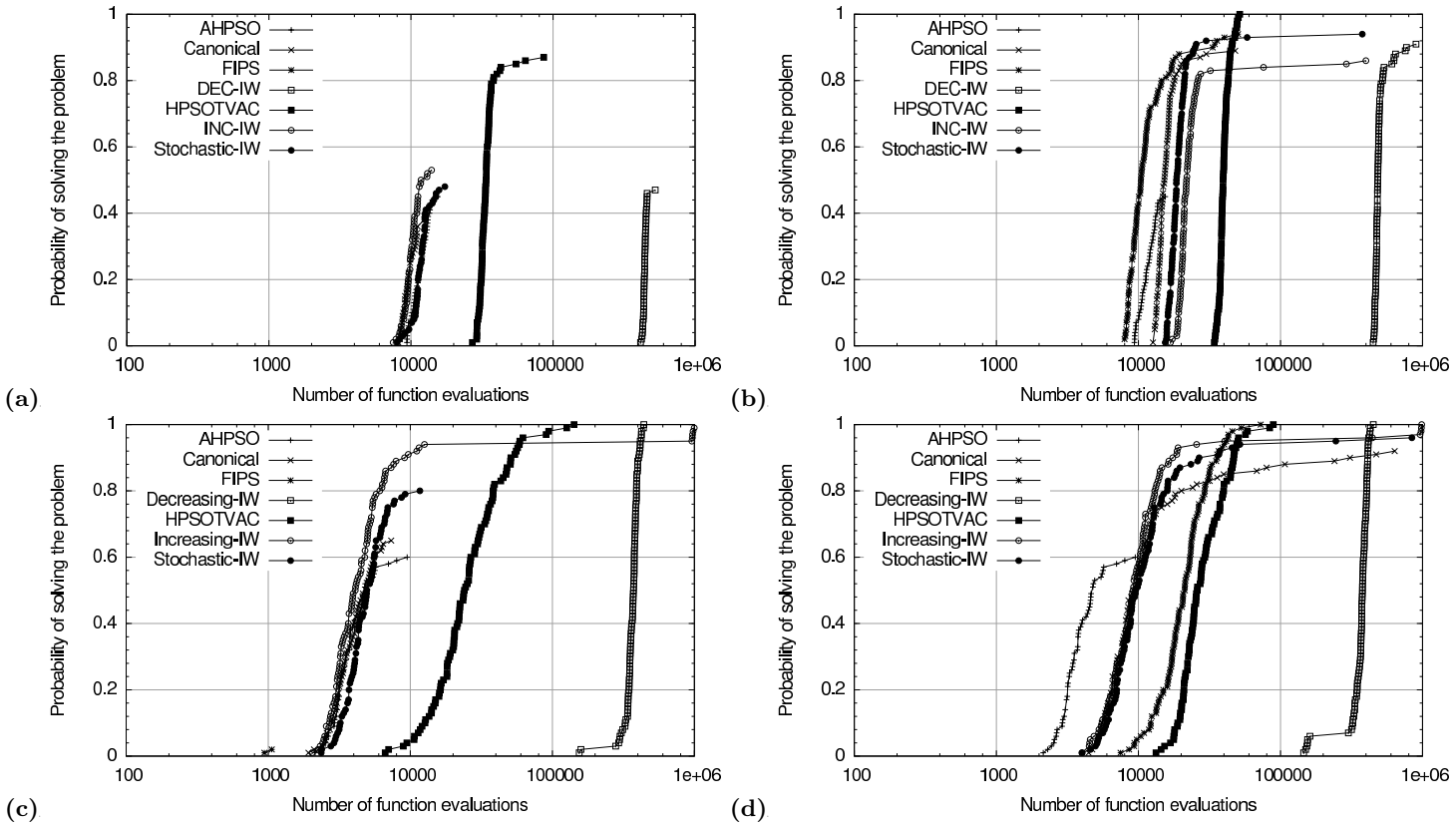

Fig. 1. Run-length distributions. In (a) and (b), the results obtained in Griewank function. In (c) and (d), the ones in Rastrigin

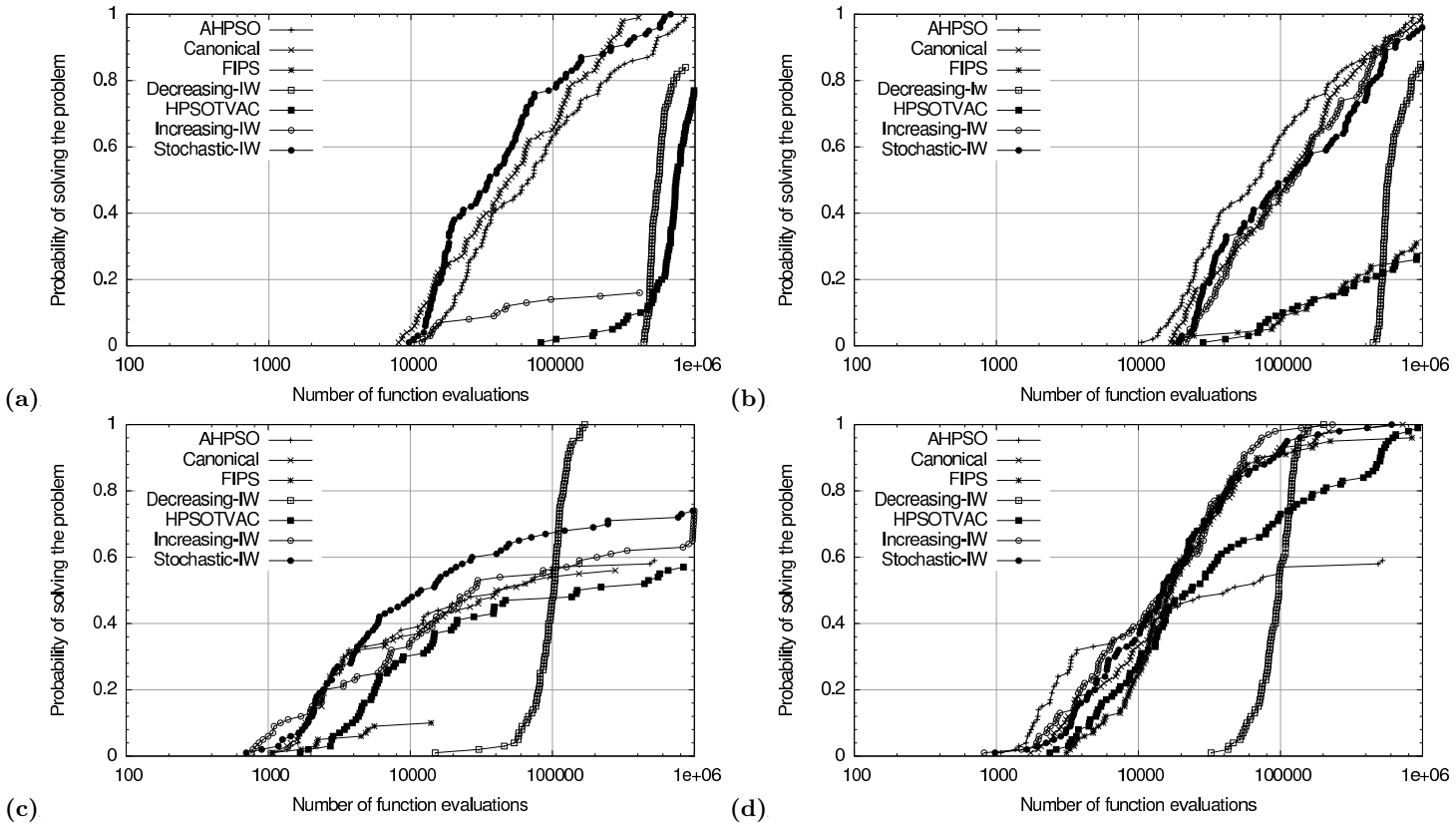

Fig. 2. Run-length distributions. In (a) and (b), the results obtained in Rosenbrock function. In (c) and (d), the ones in Schaffer's F6

Another related symptom of stagnation can be seen in the RLDs for Rosenbrock and Schaffer's F6 functions. In these cases, the RLDs have a low steepness which highlights the lack of diversification strategies in most of the algorithms. In these cases stagnation exists but is not as severe as in Griewank and Rastrigin.

The only variant that do not follow the pattern in these two problems is the one based on a decreasing inertia weight and is the only one designed with diversification in mind. This variant was designed to explore the search space at the beginning and intensify the search near the end of a run. This could explain the steepness of its RLDs in these two problems.

5 Conclusions

In this paper we empirically compared seven of the most influential or promising variants of the original particle swarm optimization algorithm. Our approach was to use run-length distributions (RLDs) and statistics of the solution quality development over time.

Regarding the behavior shown by the tested PSO variants, it is evident how important is the choice of the neighborhood topology in the performance of PSO algorithms. This is something already known in the field, but the measurement of its influence in the stagnation behavior of PSO algorithms had never been done before. With respect to our initial motivation, we limited ourselves to the comparison of some of the most influential variants, and from our results we did not find any dominant variant.

One of the advantages of RLDs is that they allow the evaluation of a stochastic optimization algorithm regardless of the actual application scenario it may be used in. Another advantage is that they allow the identification of some strengths and weaknesses of the studied algorithms that can be used to improve their performance. Future research will focus on exploiting the information provided by RLDs to the *engineering* of PSO variants. We sketched how this could be done.

Acknowledgments. This work was supported by the ANTS project, an *Action de Recherche Concertée* funded by the Scientific Research Directorate of the French Community of Belgium. Marco Montes de Oca acknowledges support from the Programme Alβan, the European Union Programme of High Level Scholarships for Latin America, scholarship No. E05D054889MX. Thomas Stützle and Marco Dorigo acknowledge suport from the Belgian National Fund for Scientific Research (FNRS), of which they are a Research Associate and a Research Director, respectively.

References

1. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, IEEE Press (1995) 1942–1948

2. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the 6th International Symposium on Micro Machine and Human Science, Piscataway, NJ, IEEE Press (1995) 39–43
3. Clerc, M., Kennedy, J.: The particle swarm–explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation **6**(1) (2002) 58–73
4. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: Proceedings of the 1998 IEEE World Congress on Computational Intelligence, Piscataway, NJ, IEEE Press (1998) 69–73
5. Kennedy, J.: Stereotyping: Improving particle swarm performance with cluster analysis. In: Proceedings of the 2000 IEEE Congress on Evolutionary Computation, Piscataway, NJ, IEEE Press (2000) 1507–1512
6. Fan, H.: A modification to particle swarm optimization algorithm. Engineering Computations **19**(8) (2002) 970–989
7. Mendes, R., Kennedy, J., Neves, J.: The fully informed particle swarm: Simpler, maybe better. IEEE Transactions on Evolutionary Computation **8**(3) (2004) 204–210
8. Hoos, H.H., Stützle, T.: Stochastic Local Search: Foundations and Applications. Morgan Kaufmann Publishers, San Francisco, CA, USA (2004)
9. Eberhart, R., Shi, Y.: Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the 2000 IEEE Congress on Evolutionary Computation, Piscataway, NJ, IEEE Press (2000) 84–88
10. Trelea, I.C.: The particle swarm optimization algorithm: Convergence analysis and parameter selection. Information Processing Letters **85**(6) (2003) 317–325
11. Shi, Y., Eberhart, R.: Empirical study of particle swarm optimization. In: Proceedings of the 1999 IEEE Congress on Evolutionary Computation, Piscataway, NJ, IEEE Press (1999) 1945–1950
12. Shi, Y., Eberhart, R.: Parameter selection in particle swarm optimization. In: Proceedings of the 7th International Conference on Evolutionary Programming VII, LNCS Vol. 1447. Springer-Verlag, New York (1998) 591–600
13. Zheng, Y.L., Ma, L.H., Zhang, L.Y., Qian, J.X.: On the convergence analysis and parameter selection in particle swarm optimization. In: Proceedings of the 2003 IEEE International Conference on Machine Learning and Cybernetics, Piscataway, NJ, IEEE Press (2003) 1802–1807
14. Zheng, Y.L., Ma, L.H., Zhang, L.Y., Qian, J.X.: Empirical study of particle swarm optimizer with an increasing inertia weight. In: Proceedings of the 2003 IEEE Congress on Evolutionary Computation, Piscataway, NJ, IEEE Press (2003) 221–226
15. Eberhart, R., Shi, Y.: Tracking and optimizing dynamic systems with particle swarms. In: Proceedings of the 2001 IEEE Congress on Evolutionary Computation, Piscataway, NJ, IEEE Press (2001) 94–100
16. Ratnaweera, A., Halgamuge, S.K., Watson, H.C.: Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Transactions on Evolutionary Computation **8**(3) (2004) 240–255
17. Janson, S., Middendorf, M.: A hierarchical particle swarm optimizer and its adaptive variant. IEEE Transactions on Systems, Man and Cybernetics–Part B **35**(6) (2005) 1272–1282
18. Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical Report 2005005, Nanyang Technological University, Singapore and IIT Kanpur, India (2005)