9th HSTAM International Congress on Mechanics
Limassol, Cyprus, 12 – 14 July, 2010

EXAMINATION OF THE PERFORMANCE OF PSO ALGORITHM WITH TIME-VARYING POPULATION

C. K. Dimou[1*], A. E. Charalampakis[1]

[1] National Technical University of Athens
Zografou Campus 15780
{ckdimou, achar}@central.ntua.gr
*(corresponding author)

Keywords: Particle Swarm Optimization, Time-Varying Population Schemes, Saw-Tooth Oscillation Scheme, Benchmark Functions.

Abstract. *Although an increase of population size usually improves the average performance of the Particle Swarm Optimization (PSO) algorithm, in some cases, however, it can become detrimental on robustness, or it can result to high computational cost. In this study, the relative performance of various PSO variants implementing time-varying population schemes is examined against the PSO algorithm with inertial parameter. The variation of the population is based on the Saw – Tooth oscillation scheme, implemented within various Evolutionary Algorithms (EAs) for demanding optimization problems. The performance is examined for a wide selection of unimodal and multimodal functions.*

1 INTRODUCTION

Particle Swarm Optimization (*PSO*) is a population-based stochastic optimization technique suitable for global optimization with no need for direct evaluation of gradients. The method, introduced by Kennedy and Eberhart[1], mimics the social behavior of flocks (swarms) of birds (particles) and satisfies the five axioms of swarm intelligence, i.e., proximity, quality, diverse response, stability and adaptability[4], which are essential for the class of connectionist models like the *PSO*[4]. The algorithm searches the Design Space (*DS*) by adjusting the trajectories of individuals, called "particles", viewed as moving points in the *DS*. These "particles" are attracted towards the positions of both their personal best solution and the best solution of the swarm in a stochastic manner[5]. Herein, two time-varying population schemes are examined and their effects on performance are discussed.

2 PARTICLE SWARM OPTIMIZATION

In *PSO*, each particle of the population occupies a given position which reflects a candidate design in the *DS*. The position of the particle is updated using the information of the velocity vector, where for simplicity the time step is considered as unity. The velocity at the next time step is a function of (a) the current velocity of the particle, (b) the position of the current best solution found by the particle and (c) the position of the current best solution found by the whole swarm, as follows[5]:

$$\upsilon_{i,j;t+1} = w \cdot \upsilon_{i,j;t} + c_1 \cdot r_1 \cdot \left(p_{i,j} - x_i\right) + c_2 \cdot r_2 \cdot \left(g_i - x_i\right), \quad \left\|\boldsymbol{\upsilon}_{j;t+1}\right\| \leq \upsilon_{\max} \tag{1}$$

where $\upsilon_{i,j;t}$ is the i^{th} velocity component of the j^{th} particle at time step t; w is the inertia coefficient, controlling the influence of the current velocity; c_1 is the cognitive parameter, controlling the influence of the best solution found by the particle; c_2 is the social parameter, controlling the influence of the best solution found by the swarm; r_1 and r_2 are uniformly distributed random values in the range [0,1]; $p_{i,j}$ is the i^{th} component of the best position encountered by the j^{th} particle until time step t; g_i is the i^{th} component of the best position encountered by the whole swarm until time step t; υ_{max} is the maximum allowable velocity of the particle and $x_{i,j;t}$ is the i^{th} component of the position of the j^{th} particle at time step t. The position of the particle at time-step t+1 is evaluated as[1]:

$$\mathbf{x}_{j;t+1} = \mathbf{x}_{j;t} + \boldsymbol{\upsilon}_{j;t+1} \tag{2}$$

where the $\mathbf{x}_{j;t}$ and $\mathbf{x}_{j;t+1}$ are the position vectors of the j^{th} particle at time step t and t+1, respectively, and $\boldsymbol{\upsilon}_{j;t+1}$ is the velocity vector of the j^{th} particle for the transition from t to t+1.

3 TIME VARYING POPULATION SCHEMES – RANKING SCHEMES

In the literature it is reported that an increase of population size improves the average performance of the *PSO* algorithm[2]. In some cases, however, it could prove detrimental on robustness or it could result to high computational cost[2]. In this work, two time-varying population schemes are examined, which are defined in the spirit of Saw – Tooth oscillation scheme[3]. The first scheme is based on exponential law, where the population is doubled/halved after one time-period (T_{period}). The second scheme is based on linear law, where the population is increased/decreased by a specific amount after one time-period. For each scheme two variants are examined. The first variant (the decreasing scheme), considers a case where the population size decreases until a minimum threshold value. At that point, the population size is restored to its initial value and the process is repeated. The second variant (the increasing scheme), considers a population where the size increases until a maximum threshold value. Again, at that point, the population size is restored to its initial value and the process is repeated.

The main idea behind the implementation of the increasing scheme is based on the simple observation that *PSO* is part of a sub-family of Evolutionary Algorithms (*EAs*), where the driving force behind optimization is adaptation of the population to suite their "environment" and communication of information among the particles of the swarm[4]. This is not the case in *EAs* like the Genetic Algorithms, where the process of natural selection benefits from a large genetic pool, the crossover operator combines only the available information from a subset of the population (the "parents") and the mutation operator arbitrarily modifies the solutions that will replace these "parents" into the next generation (the "offsprings"). Motivated by the aforementioned observations, the notion of "scouting" is introduced via the increasing scheme, where part of the swarm (the "scouts") initially explores the *DS* and the remaining part of the swarm benefits from the information gathered by these scouts and focuses its attention at the areas of interest.

The first (exponential) scheme with decreasing population is dubbed Exponential Decrease Scheme (*EDS*). In this, the population size at $t=1$ is given as:

$$N\left(t=1\right)=N_{pop,\max}=N_{pop,c}\cdot 2^{N_{steps}} \tag{3}$$

where $N_{pop,max}$ is the maximum size of the swarm and $N_{pop,c}$ is the minimum threshold value. The size of the swarm remains constant for a time span equal to T_{period}. When this is completed, the swarm size is updated following the rule of either reducing the population of time step t by half or setting the $N(t+1)=N_{pop,max}$ if $N(t+1)<N_{pop,c}$. The update rule is given as:

$$\left\{ if\ \bmod\left(\frac{t}{T_{period}}\right)=0 \quad then \left\{ \begin{array}{c} if\ 0.5\cdot N\left(t\right)\geq N_{pop,c}\ then \\ N\left(t+1\right)=0.5\cdot N\left(t\right)\ else\ N\left(t+1\right)=N_{pop,\max} \end{array} \right\} \quad else\ N\left(t+1\right)=N\left(t\right) \right\} \tag{4}$$

From eq. (4) it can be seen that the process repeats itself every $(N_{steps}+1)\cdot T_{period}$ time steps. These steps define an epoch. The T_{period} is a function of υ_{max} and parameter N_{revol}, which defines the number of revolutions a particle can travel in the *DS* at $\upsilon=\upsilon_{max}$. The T_{period} is given as:

$$T_{period}=\frac{N_{revol}}{\upsilon_{\max}}\cdot\left\|DS\right\|,\quad \left\|DS\right\|=\sqrt{\sum_{i=1}^{n}\left(x_{i,\max}-x_{i,\min}\right)^{2}} \tag{5}$$

where $x_{i,max}$ and $x_{i,min}$ are the maximum and minimum value of the i^{th} Design Variable (*DV*), $\|DS\|$ is the Euclidian distance of the main diagonal of the *DS* and n is the number of *DVs*.

The exponential scheme with increasing population is dubbed Exponential Increase Scheme (*EIS*), where the population size at $t=1$ and its update rule are given as:

$$N\left(t=1\right)=N_{pop,c} \tag{6}$$

$$\left\{ if\ \bmod\left(\frac{t}{T_{period}}\right)=0 \quad then \left\{ \begin{array}{c} if\ 2\cdot N\left(t\right)\leq N_{pop\max}\ then \\ N\left(t+1\right)=2\cdot N\left(t\right)\ else\ N\left(t+1\right)=N_{pop,c} \end{array} \right\} \quad else\ N\left(t+1\right)=N\left(t\right) \right\} \tag{7}$$

The second (linear) scheme with decreasing population is dubbed Linear Decrease Scheme (*LDS*). In this, the population size at $t=1$ and its update rule are given as:

$$N\left(t=1\right)=N_{pop,\max}=N_{pop,c}\cdot N_{steps} \tag{8}$$

$$\left\{ if \ \bmod\left(\frac{t}{T_{period}}\right)=0 \quad then \left\{ \begin{matrix} if \ N(t)-N_{pop,c} \geq N_{pop,c} \ then \\ N(t+1)=N(t)-N_{pop,c} \ else \ N(t+1)=N_{pop,\max} \end{matrix} \right\} \quad else \ N(t+1)=N(t) \right\} \tag{9}$$

The process repeats itself every $N_{steps}{\cdot}T_{period}$ time steps. The T_{period} is defined as in eq. (5).

Finally, for the linear scheme with increasing population (Linear Increase Scheme – *LIS*), the population size at t=1 is given as in eq. (6) and its update rule is given as:

$$\left\{ if \ \bmod\left(\frac{t}{T_{period}}\right)=0 \quad then \left\{ \begin{matrix} if \ N(t)+N_{pop,c} \leq N_{pop\max} \ then \\ N(t+1)=N(t)+N_{pop,c} \ else \ N(t+1)=N_{pop,c} \end{matrix} \right\} \quad else \ N(t+1)=N(t) \right\} \tag{10}$$

The variation of the swarm size for the set of parameters $\{N_{pop,c}, N_{steps}, N_{revol}, \upsilon_{max}\}=\{32, 3, 2, 10\%\|DS\|\}$ is presented in Figure 1 for the *EDS* and *EIS* schemes, and in Figure 2 for the *LDS* and *LIS* schemes. Swarm size for the *EDS* and *EIS* varies from a maximum of 256 particles to a minimum of 32 particles, whereas for the *LDS* and *LIS* swarm size varies from a maximum of 96 particles to a minimum of 32 particles.

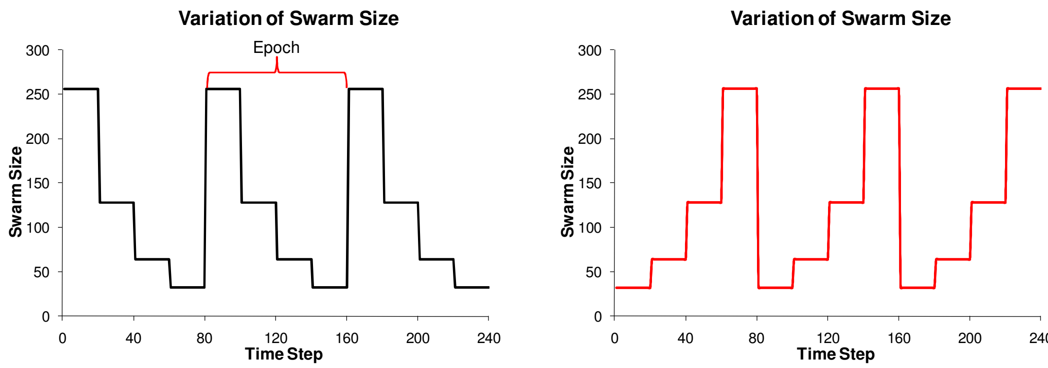

Figure 1: Swarm size variation for *EDS* and *EIS* schemes $\{N_{pop,c}, N_{steps}, N_{revol}, \upsilon_{max}\}=\{32, 3, 2, 10\%\|DS\|\}$

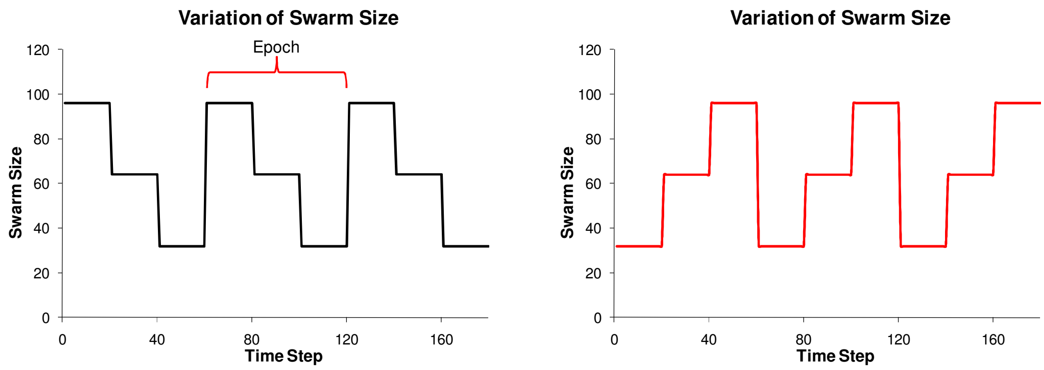

Figure 2: Swarm size variation for *LDS* and *LIS* schemes $\{N_{pop,c}, N_{steps}, N_{revol}, \upsilon_{max}\}=\{32, 3, 2, 10\%\|DS\|\}$

When the swarm size increases, the new particles are generated in random. These particles are distributed uniformly in the *DS* inheriting only the information regarding the best solution found by the swarm.

When the swarm size decreases, a process based on the relative performance of the particles is used to discard the "least performing particles". Two ranking variants are implemented, both of which are combinations of the objective of the best solution of the particle and the time elapsed from its discovery. The first variant considers as its prime criterion the best solution found by the particle. The second variant considers as its prime criterion the time elapsed from the discovery of the best solution. Thus, the latter variant promotes particles with recent discoveries of new personal bests, to enhance the explorative capacity of the algorithm.

The augmented objective used in the First Ranking Variant *(FRV)* is given as:

$$AO_j = p_j + r_j \cdot (t+1) + (t - TP_j) \tag{11}$$

where AO_j is the augmented objective of the j^{th} particle, p_j is the objective of the best solution of the j^{th} particle, r_j is the rank of the j^{th} particle with respect to its objective and TP_j is the time step where the j^{th} particle discovered its personal best solution. All quantities refer to time step *t*.

The augmented objective used in the Second Ranking Variant *(SRV)* is given as:

$$AO_j\left(t\right)=\left(t-TP_j\right)\cdot\left[\left|g\right|+\left|g_{\max}\right|+1\right]+p_j \quad g=\min_j\left\{p_j\right\} \quad g_{\max}=\max_j\left\{p_j\right\} \tag{12}$$

where g is the best objective of the personal best solutions encountered by the swarm and g_{max} is the worst objective of the personal best solutions encountered by the swarm. Eqs (11) and (12) ensure that the prime ranking criterion is always conserved. All quantities refer to time step *t*.

4 BENCHMARK FUNCTIONS

Ten benchmark functions are examined (Table 1). Functions F1 to F5 are unimodal whereas functions F6 to F10 are multimodal or extremely multimodal. For function F9, the parameters a_{ij} used in[6] are employed.

#	Name	Function	Range	Known Optima
F1	Sphere Function	$f\left(\mathbf{x}\right)=\sum_{i=1}^{n}x_i^2$	$-a\le x_i\le a \quad a=10$	$\min f(\mathbf{x})=f\left(0,...,0\right)=0$
F2	Schwefel's Double Sum Function	$f\left(\mathbf{x}\right)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2$	$-a\le x_i\le a \quad a=10$	$\min f(\mathbf{x})=f\left(0,...,0\right)=0$
F3	Generalized Rosenbrock's Function	$f\left(\mathbf{x}\right)=\sum_{i=1}^{n-1}\left[100\cdot\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$	$-a\le x_i\le a \quad a=10$	$\min f(\mathbf{x})=f\left(1,...,1\right)=0$
F4	Step Function	$f\left(\mathbf{x}\right)=\sum_{i=1}^{n}\left(\left\|x_i+0.5\right\|\right)^2$	$-a\le x_i\le a \quad a=10$	$\min f(\mathbf{x})=f\left(b,...,b\right)=0$ $b=-0.5$
F5	Quartic Function with Noise	$f\left(\mathbf{x}\right)=\sum_{i=1}^{n}i\cdot x_i^4+random\left[0,1\right)$	$-a\le x_i\le a \quad a=10$	$\min f(\mathbf{x})=f\left(0,...,0\right)=0$
F6	Ackley's Generalized Function	$f\left(\mathbf{x}\right)=\left\{\left[20-20\cdot\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)\right]+\left[e-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos\left(2\pi x_i\right)\right)\right]\right\}$	$-a\le x_i\le a \quad a=10$	$\min f(\mathbf{x})=f\left(0,...,0\right)=0$
F7	Schwefel's Generalized Function[7]	$f\left(\mathbf{x}\right)=\left\{418.9828873\cdot n-\sum_{i=1}^{n}\left[x_i\cdot\sin\left(\sqrt{\left\|x_i\right\|}\right)\right]\right\}$	$-a\le x_i\le a \quad a=512$	$\min f(\mathbf{x})=f\left(b,...,b\right)=0$ $b=420.9687$
F8	Rastrigin's Generalized Function	$f\left(\mathbf{x}\right)=\sum_{i=1}^{n}\left[x_i^2-10\cdot\cos\left(2\pi x_i\right)+10\right]$	$-a\le x_i\le a \quad a=10$	$\min f(\mathbf{x})=f\left(0,...,0\right)=0$
F9	Shekel's Foxholes Function[6]	$f\left(\mathbf{x}\right)=\left[\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}\left(x_i-a_{ij}\right)^6}\right]^{-1}$	$-65.5\le x_i\le 65.5$ $-65.5\le x_i\le 0$ $-48.5\le x_i\le 65.5$ $i=\left\{1,2\right\}$	$\min f(\mathbf{x})=f\left(-b,-b\right)\cong 1$ $b=32$
F10	Langerman's Generalized Function[8]	$f\left(\mathbf{x}\right)=-\sum_{i=1}^{n}c_i\cdot\left\{\exp\left[-\frac{1}{\pi}\sum_{j=1}^{n}\left(x_j-a_{ij}\right)^2\right]\cdot\cos\left[\pi\cdot\sum_{j=1}^{n}\left(x_j-a_{ij}\right)^2\right]\right\}$	$0\le x_j\le 10$ $a_{ij}=\left(i-1\right)\cdot n+j\in\mathbf{A}_{\mathbf{nxm}}$ $c_i=i\in\mathbf{C}_{\mathbf{n}}$	Not known

Table 1: Benchmark Functions.

5 NUMERICAL RESULTS

To investigate the performance of the proposed algorithms, a series of parametric studies is performed. With regard to the parameters of the *PSO,* the following values are considered; w={1.0, 0.8, 0.6}, c_1={1.5, 2.0, 2.5}, c_1+c_2=4.0, υ_{max}={10%||*DS*||, 15%||*DS*||, 20%||*DS*||}[9] and swarm size N={64, 128} creating a basis of 3·3·3·2=54 analyses with different parameter sets.

The four time-varying population schemes are coupled with two ranking variants, creating 8 variants dubbed as *EDS-FRV*, *EDS-SRV*, *EIS-FRV*, *EIS-SRV*, *LDS-FRV*, *LDS-SRV*, *LIS-FRV* and *LIS-SRV*.

With regard to the newly introduced parameters the following values are considered; $N_{pop,c}$={16, 32}, N_{steps}={2, 3, 4} and N_{revol}={1, 2}. These create a basis of 2·3·2=12 analyses which are coupled with the 3·3·3=27 combinations concerning values of w, c_1 and υ_{max}. For each of these eight variants 27·12=324 analyses are performed.

Each benchmark function is examined for n={2, 5, 10} where n is the number of *DV*s with the exception of F9, where different shapes of the *DS* are considered to examine the ability of the algorithm to locate the shifted global optimum (Table 1). Each analysis for the *PSO* and the variants under examination consist of 30 runs with a different initial random seed value to obtain meaningful statistics.

Due to space limitations we focus on the results for n=10. The average objective of the best solution for the *PSO* and the examined variants for all benchmark functions are presented in Figure 3. For function F9 the min and max values of the *DVs* are {x_{min}=–64.5, x_{max}=64.5}.

For the unimodal functions F1 to F5, the *PSO* manages to outperform the proposed variants after 5,000 function evaluations. This is not the case for the multimodal functions (with the exception of F9) where the variants outperform the *PSO*. For function F9 the results show that, for this function, the *PSO* as well as the examined variants are exceptionally suited optimization algorithms.

Focusing on the relative performance of the examined variants, it is observed that the *EIS* and *LIS* variants outperform their corresponding *EDS* and *LDS* variants in all cases. Additionally, the *EDS* and *EIS* variants outperform the *LDS* and *LIS* variants. Focusing on the ranking variants, it is observed that the *FRV* variant outperforms the *SRV* variant. Thus, in terms of the average objective of the best solution, the scheme based on exponential law outperforms the scheme based on linear law and the notion of “scouting” improves the robustness of the optimization schemes.

For function F10, the optimization process is extended to 20,000 function evaluations. The average objective value of the best solution found by the swarm for the standard *PSO* and the *EDS* and *EIS* variants are presented in Figure 4. For [N_{revol}, N_{steps}]=[1, 2] the *NR*1-*NS*2 denotation is adopted; similar denotations are adopted for the remaining combinations. For the *EDS* and *EIS* variants, the average objective of the best solution is obtained over 324/6=54 analyses as in the case of the standard PSO.

For all combinations, the *EDS* and *EIS* variants outperform the *PSO* at 20,000 function evaluations. It can be seen that the *PSO* exhibits signs of stagnation whereas all the examined combinations continue to improve the average. Focusing on the examined combinations, it is observed that for the *EDS* variant the best results are observed for N_{steps}≈3-4. On the other hand, for the *EIS* variant the best results are observed for $N_{steps}=2$. With regard to N_{revol}, the best results eventually are observed for $N_{revol}=2$, particularly in the *EIS* variant.

Finally, we focus our attention in the performance of the proposed variants with respect to the absolute minimum found by the optimization schemes for benchmark functions F6, F7, F8 and F10.

In Figure 5, the objective value of the best solution for the *EDS* and *EIS* variants is presented for these functions. The proposed variants outperform the *PSO* for these functions, with the exception of *EIS-FRV* at 2,000 function evaluations in the case of F7 and at 1,000 function evaluations in the case of F8.

In Figure 6, the objective value of the best solution is presented for the *EIS-FRV* combinations for F6, F7, F8 and F10. For the *EIS-FRV* combination, it is observed that only the *NR*1-*NS*4 and *NR*2-*NS*4 combinations fail to outperform the *PSO* for F10 whereas for F8 these combinations are *NR*1-*NS*4, *NR*2-*NS*4 and *NR*2-*NS*3. The best results overall are observed for the *NR*2-*NS*2 combination.

Focusing on the performance of *EIS-FRV* combination with respect to its capacity in discovering the global optimum for functions F6, F7, F8 and F10, it is observed that for function F6 the absolute min found after 20,000 function evaluations is equal to the global optimum F6(opt)=0.00 (*PSO* F6(opt)= $2.0\cdot10^{-10}$). The global optimum is found by all combinations with respect to N_{revol} and N_{steps} but *NR*1-*NS*4 and *NR*2-*NS*4. Moreover, for function F7 the absolute min found after 20,000 function evaluations is equal to F7(opt)=$2.76\cdot10^{-7}$ (*PSO* F6(opt)= $1.16\cdot10^{-6}$) which is a local minimum. This minimum is found by *NR*2-*NS*2. Furthermore, for function F8, the absolute min found after 20,000 function evaluations by the *NR*2-*NS*2 combination is F8(opt)=$5.74\cdot10^{-8}$ (*PSO* F8(opt)= $1.23\cdot10^{-2}$) although the best local optimum is found by the *EDS-SRV* variant for the *NR*2-*NS*3 combination with F8(opt)=$3.11\cdot10^{-8}$. Finally, for function F10 all the proposed variants, all combinations for the *EIS-FRV* variant and the PSO converge to F10(opt)=–1 and the optimum, in this case, is found in less than 10,000 function evaluations.

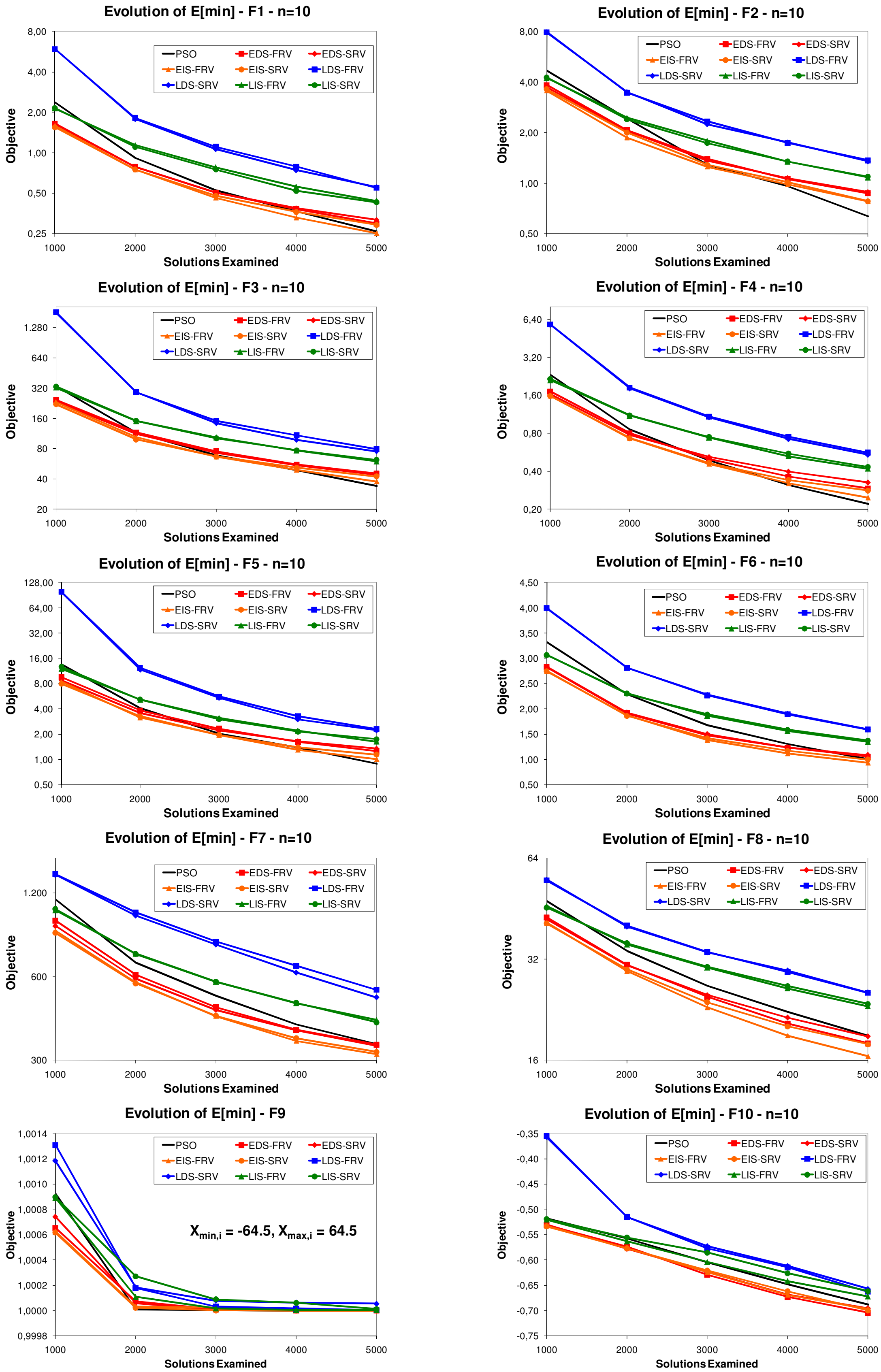

Figure 3: Evolution of E[min] functions F1 to F8 and F10, n=10, function F9, n=2 {x_{min}=−64.5, x_{max}=64.5}.

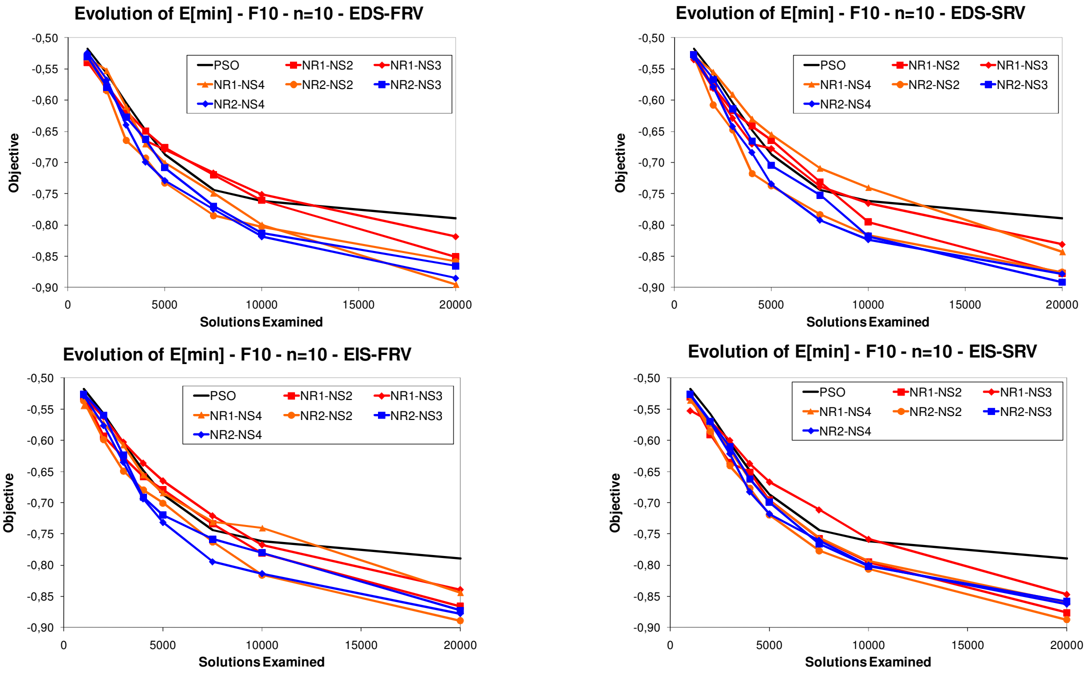

Figure 4: Evolution of E[min] for function F10, *n*=10, analysis for *EDS* and *EIS* variants.

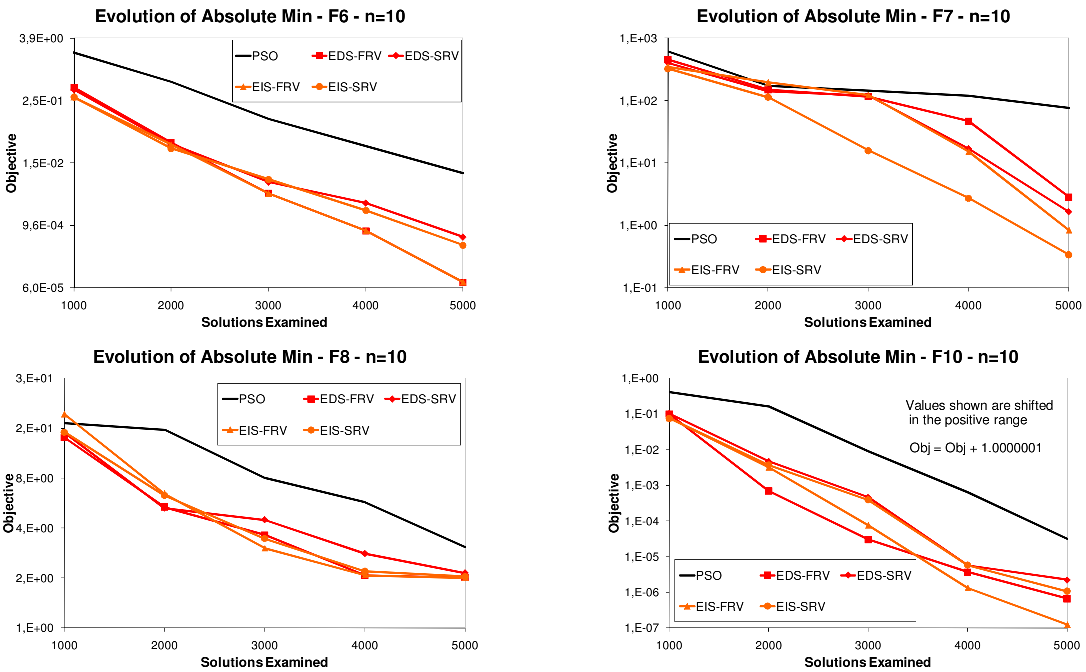

Figure 5: Objective of best solution for functions F6, F7, F8 and F10.

6 CONCLUSIONS

In this work, two time-varying population schemes are employed within the *PSO* algorithm. The first scheme is based on exponential law whereas the second one on linear law. For both schemes, two variants are considered. In the first variant (the decreasing scheme), the size decreases until a minimum-threshold value. At that point the population size is restored to its initial value and the process repeats itself. The second variant (the increasing scheme), considers a population which is increased until a maximum-threshold value.

The robustness of the time-varying population schemes and its variants is examined against the *PSO* algorithm for 10 benchmark problems. From the results, it is derived that although the *PSO* outperforms the examined variants for functions F1 to F5 (unimodal functions), the variants manage to produce consistently better results for the multimodal functions F6 to F8 and F10. The *EDS* and *EIS* variants manage to outperform their corresponding *LDS* and *LIS* variants in all cases and the *EIS* and *LIS* variants outperform the respective *EDS* and *LDS* variants. Focusing on the ranking variants it can be seen that *FRV* outperforms *SRV*.

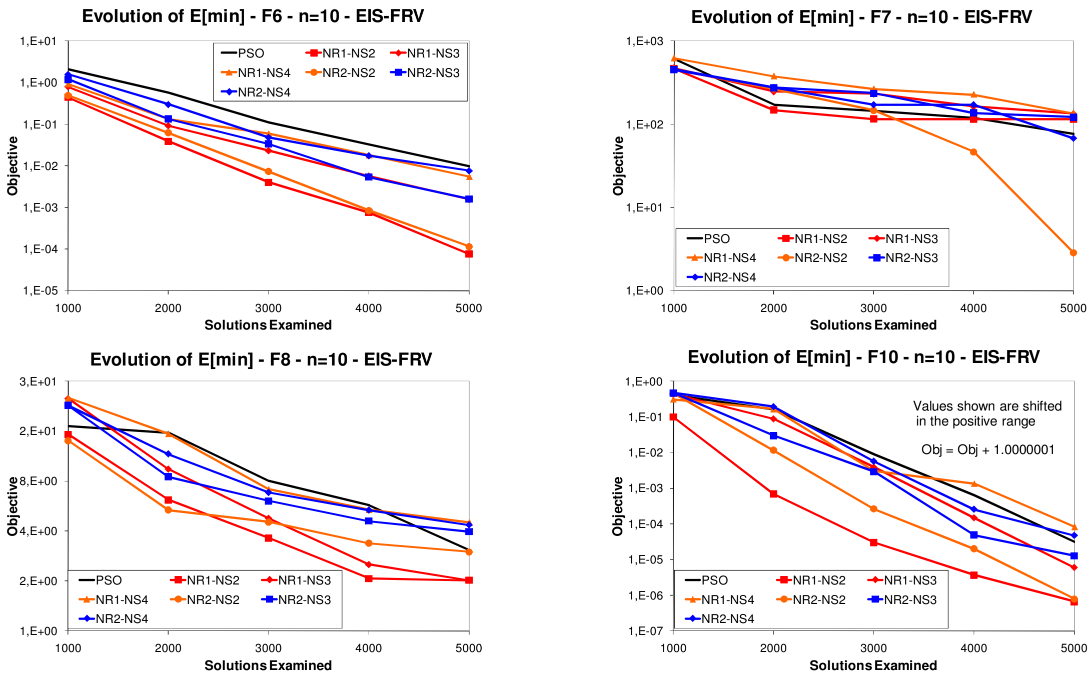

Figure 6: Objective of best solution for functions F6, F7, F8 and F10, *EIS-FRV*.

In terms of the objective of the best solution, it is observed that the proposed variants outperform the *PSO* for multimodal functions F6, F7, F8 and F10. For F6, F7 and F10 the best results are produced by the variant *EIS-FRV* for the *NR*2-*NS*2 combination.

In conclusion, it is demonstrated that the proposed time-varying population schemes (and in particular the *EIS* variant) exhibit increased performance in the case of multimodal functions. Still, the results are indicative and thus non-conclusive and further investigation is required to gain an in-depth knowledge with regard to the effects of implementing time-varying population schemes in *PSO*.

REFERENCES

[1] Kennedy J., Eberhart R.C. (1995), "Particle swarm optimization", *Proc., of IEEE Int., Conf., on Neural Networks*, Perth Australia, Vol. IV, pp. 1942-1948.

[2] Dimou C.K., Koumousis V.K., (2007), "Particle Swarm Optimization in the Reliability Based Optimal Design of Structures", *8th HSTAM International Congress of Mechanics*, Patras 12-13 July.

[3] Koumousis V.K., Katsaras C.P. (2006), "A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance", IEEE Transactions on Evolutionary Computation, 10, pp. 19-28.

[4] Millonas M.M. (1993), "Swarms, Phase Transitions and Collective Intelligence" Artificial Life III, Santa Fe Institute, Studies in the Sciences of Complexity, XVII, pp. 417-445.

[5] Clerc M., Kennedy J. (2002), "The Particle Swarm - Explosion, Stability, and Convergence in a Multidimensional Complex Space", IEEE Transactions on Evolutionary Computation, 6, pp. 58-73.

[6] Yao, X. Liu Y., Lin G. (1999), "Evolutionary Programming Made Faster", IEEE Transactions On Evolutionary Computation, 3(2), pp. 82-102.

[7] Schwefel, H.P. (1995), *Evolution and Optimum Seeking*. John Wiley & Sons.

[8] Bersini H., Dorigo M., Langerman S., Seront G., and Gambardella L.M. (1996), "Results of the first international contest on evolutionary optimisation (1st ICEO)". In *Proc. of IEEE International Conference on Evolutionary Computation, IEEE-EC 96*, Nagoya, Japan, May 20-22, pp. 611-615, IEEE Press.

[9] Eberhart, R.C., Shi, Y. (2001) "Particle swarm optimization: developments, applications, and resources". *Proceedings of IEEE Congress on Evolutionary Computation 2001*, Piscataway, N.J., pp. 81–86.