Journal of Computational Design and Engineering 6 (2019) 129–142

Contents lists available at ScienceDirect

Journal of Computational Design and Engineering

journal homepage: www.elsevier.com/locate/jcde

A PSO based approach: Scout particle swarm algorithm for continuous global optimization problems

Hasan Koyuncu *, Rahime Ceylan

Electrical & Electronics Engineering Department, Konya Technical University, Konya, Turkey

ARTICLE INFO

Article history:
Received 13 September 2017
Received in revised form 21 August 2018
Accepted 24 August 2018
Available online 27 August 2018

Keywords:
Scout particle swarm optimization
Numerical function optimization
Particle swarm optimization
Artificial bee colony optimization
Hybrid approach

ABSTRACT

In the literature, most studies focus on designing new methods inspired by biological processes, however hybridization of methods and hybridization way should be examined carefully to generate more suitable optimization methods. In this study, we handle Particle Swarm Optimization (PSO) and an efficient operator of Artificial Bee Colony Optimization (ABC) to design an efficient technique for continuous function optimization. In PSO, velocity and position concepts guide particles to achieve convergence. At this point, variable and stable parameters are ineffective for regenerating awkward particles that cannot improve their personal best position (P_{best}). Thus, the need for external intervention is inevitable once a useful particle becomes an awkward one. In ABC, the scout bee phase acts as external intervention by sustaining the resurgence of incapable individuals. With the addition of a scout bee phase to standard PSO, Scout Particle Swarm Optimization (ScPSO) is formed which eliminates the most important handicap of PSO. Consequently, a robust optimization algorithm is obtained.

ScPSO is tested on constrained optimization problems and optimum parameter values are obtained for the general use of ScPSO. To evaluate the performance, ScPSO is compared with Genetic Algorithm (GA), with variants of the PSO and ABC methods, and with hybrid approaches based on PSO and ABC algorithms on numerical function optimization. As seen in the results, ScPSO results in better optimal solutions than other approaches. In addition, its convergence is superior to a basic optimization method, to the variants of PSO and ABC algorithms, and to the hybrid approaches on different numerical benchmark functions. According to the results, the Total Statistical Success (TSS) value of ScPSO ranks first (5) in comparison with PSO variants; the second best TSS (2) belongs to CLPSO and SP-PSO techniques. In a comparison with ABC variants, the best TSS value (6) is obtained by ScPSO, while TSS of BitABC is 2. In comparison with hybrid techniques, ScPSO obtains the best Total Average Rank (TAR) as 1.375, and TSS of ScPSO ranks first (6) again. The fitness values obtained by ScPSO are generally more satisfactory than the values obtained by other methods. Consequently, ScPSO achieve promising gains over other optimization methods; in parallel with this result, its usage can be extended to different working disciplines.

1. Introduction

Optimization methods are important algorithms required in control, classification, communication and other fields. They attempt to satisfy the requirements of convergence in achieving optimal solutions. Especially in online (real-time) applications, fast and successful convergence is required.

Over time, optimization algorithms have come up short on convergence performance. At the same time, two approaches (new methods or a hybridization of some optimization methods) have been taken to obtain better results. In this study, the second option (hybridization) is pursued, and an efficient method is examined with its operation conditions for benchmark function optimization.

Particle Swarm Optimization (PSO) is effective and superior to many-newly developed optimization algorithms in different disciplines (Kennedy & Eberhart, 1995). However, standard PSO can be made more effective and applicable with the addition of another optimization algorithm, another algorithm component or inspired formulas from it. In this way, hybrid or improved PSO architectures are developed. In the literature, these modifications are required to obtain better convergence; the structures are generally evaluated using benchmark functions.

Peer review under responsibility of Society for Computational Design and Engineering.

* Corresponding author at: Konya Technical University, Faculty of Engineering and Natural Sciences, Electrical & Electronics Engineering Department, Konya, Turkey.

E-mail addresses: hasankoyuncu@selcuk.edu.tr (H. Koyuncu), rpektatli@selcuk.edu.tr (R. Ceylan).

https://doi.org/10.1016/j.jcde.2018.08.003
2288-4300/

Eberhart and Shi (2000) modified the basic PSO algorithm to obtain a Modified Particle Swarm Optimizer (Global PSO-w) which includes the parameter of inertia weight. In their study, an optimum value was found based on trials using Schaffer's f6 function. Concerning this, it was understood that the inertia weight is essential for higher performance. Clerc and Kennedy (2002) analyzed the path of a particle in discrete time and continuous time. In particular, they examined constricted coefficients (Global PSO-cf) on unconstrained-real valued benchmark functions. Thus, the importance and the effect of constricted coefficients were revealed. Kennedy and Mendes (2002) presented the topological importance of PSO (Local PSO-w, Local PSO-cf). Parsopoulos (2004) surveyed Unified Particle Swarm Optimization (UPSO) for numerical optimization and proved that UPSO is more effective than local and global PSO variants. Liang, Qin, Suganthan, and Baskar (2006) used the historical best information of all particles to update the velocity and obtained a Comprehensive Learning Particle Swarm Optimizer (CLPSO). They also compared this method with PSO based algorithms on multimodal test functions. Shin and Kita (2014) improved PSO by using second personal best and second global best particles in addition to global best and personal best values. As a result, they obtained Second Personal-PSO (SP-PSO) and Second Global-PSO (SG-PSO) structures. These algorithms were compared with Global PSO-w, Global PSO-cf, Local PSO-w, Local PSO-cf, UPSO and CLPSO on numerical benchmark functions. Their experiments showed that SP-PSO was a robust and adequate algorithm providing better convergence.

Mendes, Kennedy, and Neves (2004) modified the Canonical Particle Swarm algorithm and designed the Fully Informed Particle Swarm in which all individuals were fully informed. They also examined different topologies of Fully Informed PSO for numerical function optimization. According to their study, ring and square topologies achieved better results, and Fully Informed PSO outperformed Canonical PSO on benchmark function optimization. Zhan, Zhang, Li, and Shi (2011) proposed an orthogonal learning (OL) strategy for PSO in which the OL strategy allowed the particles to fly in better directions. The OL strategy was also applied to both global and local versions of PSO. Consequently, the OLPSO-G and OLPSO-L algorithms were produced. Based on the results, OLPSO-L owned to the best performance among its peers on benchmark function optimization. Lukasik and Kowalski (2014) compared some effective techniques (Fully Informed PSO, the Firefly algorithm and Glowworm Swarm Optimization) on continuous function optimization. They proved that Fully Informed PSO was the most effective and least-computationally expensive method.

In addition to PSO variants, ABC based algorithms have a wide range of use with regards to obtaining higher convergence. Karaboga (2005) designed the ABC algorithm which is a stochastic optimization method similar to PSO, but ABC utilizes the foraging behaviours of honey bees to solve optimization problems. According to our experiments, the convergence of ABC is reliable, but the speed of convergence is slow. ABC also has some limitations when operating with some multimodal functions, and its stochastic nature slows down the convergence speed (Karaboga & Akay, 2009). The reason for this is an improper balance between exploration and exploitation (Zhu & Kwong, 2010). ABC is good at exploration but poor at exploitation (Zhu & Kwong, 2010). In the literature, handicaps (convergence speed, the trade-off between exploration-exploitation, and convergence performance (arrival to desired value)) were bypassed and the capability of ABC was upgraded. Pampara and Engelbrecht (2011) proposed three kind of binary based ABC algorithms: binary ABC (BinABC), angle modulated ABC (AMABC) and normalized ABC (normABC). These algorithms were similar to ABC, but they needed to be transformed from continuous space to discrete space, which translates into high complexity. In Pampara and Engelbrecht (2011), techniques were compared on unconstrained benchmark functions. Kashan, Nahavandi, and Kashan (2012) designed a new artificial bee colony algorithm (DisABC) for binary optimization. Differential expression of this technique was executed in continuous space. At this time, it is used in a two-phase heuristic to construct a complete solution in binary space (Kashan et al., 2012). While doing this, Jaccard's coefficient of similarity/dissimilarity was used. For performance measurements, DisABC was tested on benchmark function optimization. Jia, Duan, and Khan (2014) generated a binary based ABC algorithm (BitABC) using a bitwise operation for the movements of employed and onlooker bees. In their study, BitABC was compared with normABC, BinABC, DisABC and with Genetic Algorithm (GA) on benchmark functions. Their results revealed that BitABC was generally superior to the other methods and to GA with regards to convergence performance. Karaboga and Gorkemli (2014) designed the Quick Artificial Bee Colony (qABC) algorithm which is a new version of the ABC algorithm. In their study, qABC simulated the behaviour of onlooker bees more accurately, and the performance of ABC was improved in terms of local searching ability.

There are some studies touching on the hybridization of PSO and ABC algorithms. El-Abd (2011) used the update formula of the employed bee phase (also the same as in the onlooker bee phase) in the general PSO algorithm for continuous function optimization, and named the algorithm as $Hybrid_{FP}$. According to the experiments, the performance of $Hybrid_{FP}$ was remarkable in comparison with ABC and PSO algorithms in terms of achieving better results on unimodal functions. However, the performance of ABC and other algorithms was better than $Hybrid_{FP}$ on multimodal functions. Wang, Lv, Zhao, and Zhang (2012) designed a hybrid approach to solve complex-constraint optimization problems, and formed the structure based on this aim. The PSO algorithm was utilized to choose the food sources for ABC. According to the trials, the proposed PSO-ABC approach outperformed other approaches, and reached better solutions. Yang and Pei (2013) optimized the parameters of the PSO algorithm by using ABC, and named the hybrid approach as HAP. According to the results, the proposed method obtained better results than other methods on travelling salesman problem. Kiran and Gunduz (2013) utilized the global best solution of PSO and the best solution of ABC in order to form a more robust optimization algorithm. Hybrid Approach based on PSO and ABC (HPA) was tested on continuous optimization problems. Concerning this, remarkable convergence was obtained by HPA in comparison with other algorithms. Chun-Feng, Kui, and Pei-Ping (2014) suggested a hybrid technique (ABC-PS) based on ABC in which PSO is processed to find new solutions. According to the results, ABC-PS outperformed the variants of ABC in global optimization. Li, Wang, Yan, and Li (2015) combined the local search in PSO with onlooker bee and scout bee phases in ABC to solve high dimensional optimization problems. According to the trials, the proposed method (PS-ABC) achieved better performance than peer approaches on high dimensional problems.

In this study, we improved the PSO algorithm with a necessary part, the scout bee phase in ABC, which eliminated the most important handicap of PSO. In contrast to other methods found in the literature, the hybridization of techniques is realized based on the need of the PSO algorithm. As a result, we can prevent the hybrid method from becoming cumbersome. A performance analysis of the proposed method (Scout Particle Swarm Optimization-ScPSO) is examined in unimodal, multimodal, separable, partially-separable, non-separable, differentiable, non-differentiable, scalable, shifted and rotated situations. A convergence analysis of ScPSO was performed on continuous function optimization. In addition, the optimum operating conditions of ScPSO are

investigated, and the method is compared with variants of PSO, with variants of ABC and with hybrid approaches based on PSO and ABC methods that have proven to be efficient at global numerical optimization.

To determine ScPSO parameters, general PSO parameters and the parameter *limit*, are investigated over a wide range in which the best convergence could be found. Hence, parameter selection of ScPSO is defined for general use in various disciplines.

The ScPSO algorithm is first compared with Global PSO-w, Global PSO-cf, Local PSO-w, Local PSO-cf, UPSO, CLPSO, SG-PSO and SP-PSO on 8 numerical benchmark functions. Second, ScPSO is compared with GA and with ABC based techniques (normABC, BinABC and DisABC) on 8 benchmark functions. Third, ScPSO is compared with the PSO- ABC based hybrid algorithms on 8 numerical benchmark functions. Thus, this is an attempt to determine whether ScPSO is superior to other methods or not on global function optimization.

2. Methods

The choice of optimization method plays an important role in convergence, and the performance of these methods differs depending on their approaches and formulas. In some cases, the parameters and formulas can be deficient at achieving better convergence.

Particle Swarm Optimization includes a handicap, which is the absence of the regeneration of ineffective particles that cannot improve their P_{best} values. On the other hand, the ABC algorithm contains a scout bee phase to eliminate the handicap of regeneration. For this reason, we added the scout bee phase into Standard PSO to upgrade its performance (Koyuncu & Ceylan, 2015). As a result, a substantial structure (Scout Particle Swarm Optimization) has been developed. In the study of Koyuncu and Ceylan (2015); ScSPO was investigated for pattern classification and was compared only with the basic PSO algorithm. Additionally, the optimal operating conditions of ScPSO are unknown, and it is not clear whether ScPSO is sufficient to challenge prior variants and hybrid versions of PSO and ABC algorithms or not.

In this study, optimal operating conditions of ScPSO are investigated using continuous function optimization. A convergence speed comparison including the iteration number is applied to ScPSO, and a three step evaluation strategy is used to obtain optimal parameter values and ranges. ScPSO is then compared with various algorithms on the convergence of numerical benchmark problems to assess its performance. In this section, PSO and ABC algorithms are first explained and then, ScPSO is examined in detail.

2.1. Particle Swarm optimization

Particle Swarm Optimization is a swarm based algorithm inspired by the foraging behaviours of birds (Ceylan & Koyuncu, 2016). During the imitation of foraging behaviours, PSO utilizes the velocity and position phenomena for the update of its particles. Additionally, the previous particle information is used in general backpropagation processes (Koyuncu & Ceylan, 2013). In this manner, a continuous progression towards the global optimum point is provided. The flowchart of PSO is shown in Fig. 1 (Koyuncu & Ceylan, 2015).

For the velocity update, some constants and variables that are used are shown in Eq. (1): w (inertia weight), $V_i(t)$ (former velocity), c_1 and c_2 (acceleration constants), r_1 and r_2 (random values within [0,1]), $X_i(t)$ (former position), $X_{pbest(i)}(t)$ (individual best position of i.th particle) and $X_{gbest}(t)$ (global best position of the whole swarm). For the new position value, the old position ($X_i(t)$) and new velocity ($V_i(t)$) values are summed as in Eq. (2).

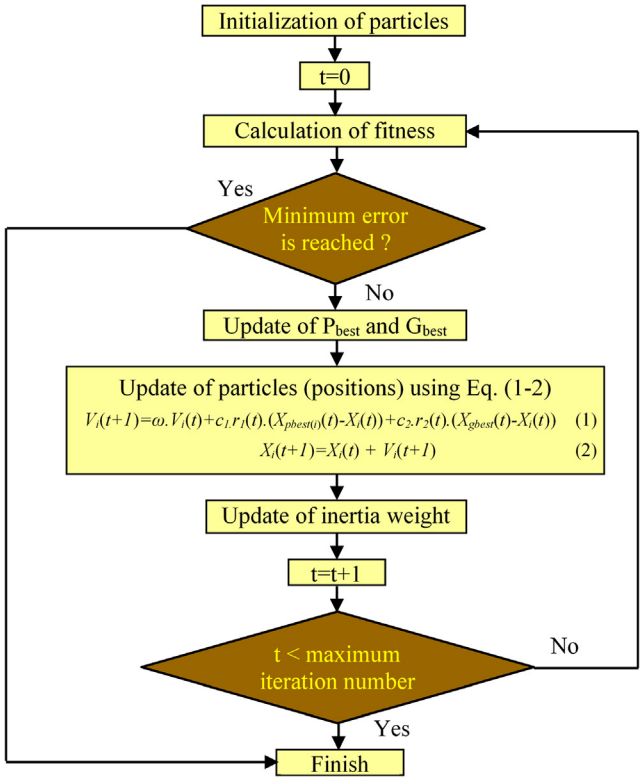

Fig. 1. Flowchart of PSO algorithm.

$$V_i(t+1) = \omega V_i(t) + c_1 r_1 \left(X_{pbest(i)}(t) - X_i(t)\right) + c_2 r_2 \left(X_{gbest}(t) - X_i(t)\right) \quad (1)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (2)$$

The operation of PSO can be divided into 2 parts: an initialization part and loop part. The loop part can be subdivided into 4 parts: calculation of fitness, update of P_{best} and G_{best} values, update of positions and update of the inertia weight.

2.2. Artificial bee Colony optimization

Artificial Bee Colony Optimization is a stochastic optimization algorithm simulating the actions of honey bees divided into three groups: employed bees, onlooker bees and scout bees (Karaboga, 2005).

Employed bees act as the pioneers to onlooker bees. These bees fly to the food source, carry it to hive, and perform the waggle dance in the dance area. At this time, the waggle dance directs the onlooker bees to the food source (Koyuncu & Ceylan, 2015). The onlooker bees are seen as the observers in the dance area. To find the food source, onlooker bees study the movements of employed bees. Scout bees are the saviors, starting a random search for food after the food source is consumed.

The pseudocode of ABC is as follows:

1. Initialization Section

-Generate the food sources (x_m) using Eq. (3)

($l_i$is the lower and u_i is the upper limit of the space)

$$x_m = l_i + rand(0,1) * (u_i - l_i) \quad (3)$$

While (iteration number < maximum iteration number)

2. Employed Bee Section

-Generate the neighboring food sources (v_{mi}) using Eq. (4) for all food sources

(v_{mi} vectors are going to be compared with x_m vectors)

$$v_{mi} = x_{mi} + \phi_{mi}(x_{mi} - x_{ki}) \tag{4}$$

• x_{ki} is a randomly chosen food source, and ϕ is a random number within the range [−1,1]

• i is a randomly chosen index within the range [1,dimension number]

• k can't have the same value as the parameter m

-Calculate the fitness function according to Eq. (5)

$$fit_m(x_m) = \begin{cases} 1/(1+f_m(x_m), & f_m(x_m) > 0 \\ 1 + |f_m(x_m)|, & f_m(x_m) < 0 \end{cases} \tag{5}$$

-For m = 1:population size

If ($fit_m(v_{mi})$ is better than $fit_m(x_m)$) then $x_m = v_{mi}$,

Else $x_m = x_m$ and $count(m)=count(m)+1$;End;End

3. Onlooker Bee Section

-Calculate the profitability of all food sources (x_m vectors) using Eq. (6).

$$P_m = \frac{fit_m(x_m)}{\sum_{m=1}^{SN} fit_m(x_m)} \tag{6}$$

- z = 0; m = 1;

While (z < population size)

If (P_m > rand (0, 1))

--Generate the new neighbor food source (y_{mi}) using Eq. (4)

--Calculate its fitness using Eq. (5)

If ($fit_m(v_{mi})$ is better than $fit_m(x_m)$) then $x_m = y_{mi}$

Else $x_m = x_m$ and $count(m)=count(m)+1$

-- z=+1; End

Else m = m + 1 (if m exceeds the population size, then m = 1); End; End

4. Scout Bee Section

- For m = 1:population size

If ($count(m)$ > user defined *limit*), then use Eq. (3) to reproduce x_m,

End

End

In ABC, the employed bee number is equal to half of the swarm and other half consists of onlookers. The employed bee whose food source runs out becomes a scout bee.

As seen in the pseudocode, the employed bee phase operates Eqs. (4) and (5) to generate new particles for comparison with old ones. The onlooker bee phase uses Eqs. (4)–(6) with the same purpose as the employed bee phase. However, Eq. (6) provides a probability choice which contributes to reasonable diversity. The scout bee phase utilizes Eq. (3) in order to regenerate the useless particles. For this purpose, this phase operates a user defined parameter called *limit*.

2.3. Scout particle Swarm optimization

In the PSO algorithm, velocity and position are responsible for the update of particles. To do this, ABC uses employed bee and onlooker bee phases. Nevertheless, the scout bee phase in ABC regenerates useless particles that cannot improve their individual best positions. As explained above, this process is operated via the parameter *limit*.

The Standard PSO algorithm (explained in Section 2.1 doesn't contain a control parameter to regenerate insufficient particles. At this point, these particles are the ones that cannot retrieve their P_{best} value. Particles are updated without any diversity in Standard PSO, and their adequacies are not controlled (Koyuncu & Ceylan, 2015). It is obvious that PSO needs a control parameter to improve its convergence capability, but this parameter must not increase its convergence time significantly. Consequently, it looks similar to a reasonable idea to insert the scout bee phase into the Standard PSO algorithm. By adding the scout bee phase into PSO, ScPSO is obtained (Koyuncu & Ceylan, 2015). In ScPSO, all processes (except *limit*) are the same with the PSO algorithm. The flowchart of ScPSO is shown in Fig. 2 (Koyuncu & Ceylan, 2015).

As seen in Fig. 2, the particle, which can't improve its personal best position (P_{best}), is regenerated according to Eq. (3). There, q_i and l_i symbolize the upper and the lower boundaries of particle, respectively. In the regeneration process, q_i and l_i are equal to particle's maximum position X_{max} and to particle's minimum position X_{min}.

In Fig. 2, minimum error is the difference between the global point, the value to be reached, and the value obtained by continuous global optimization. For a real world problem, this situation changes depending on the main purpose and usage of the ScPSO. In the study of Koyuncu and Ceylan (2015), the minimum error symbolizes the Mean Square Error (MSE), since the aim and the usage are respectively the pattern classification and optimization of the Neural Network (NN) algorithm. In brief, minimum error can be defined depending on the application, the methods utilized and the purpose of the usage.

The inspiration of ScPSO is presented in Fig. 3 to better explain the formation of hybrid technique (Ceylan & Koyuncu, 2018).

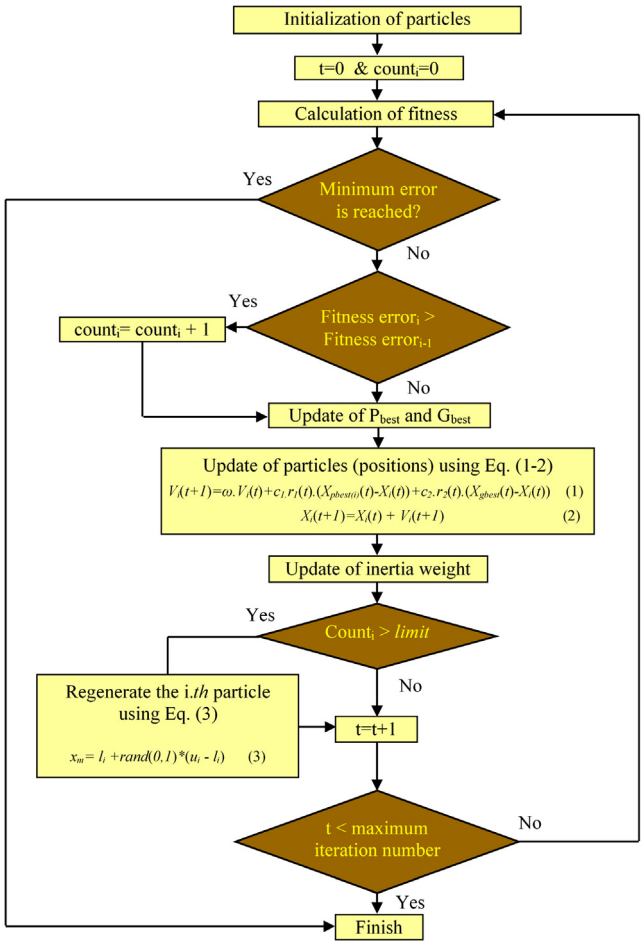

Fig. 2. Flowchart of ScPSO algorithm.

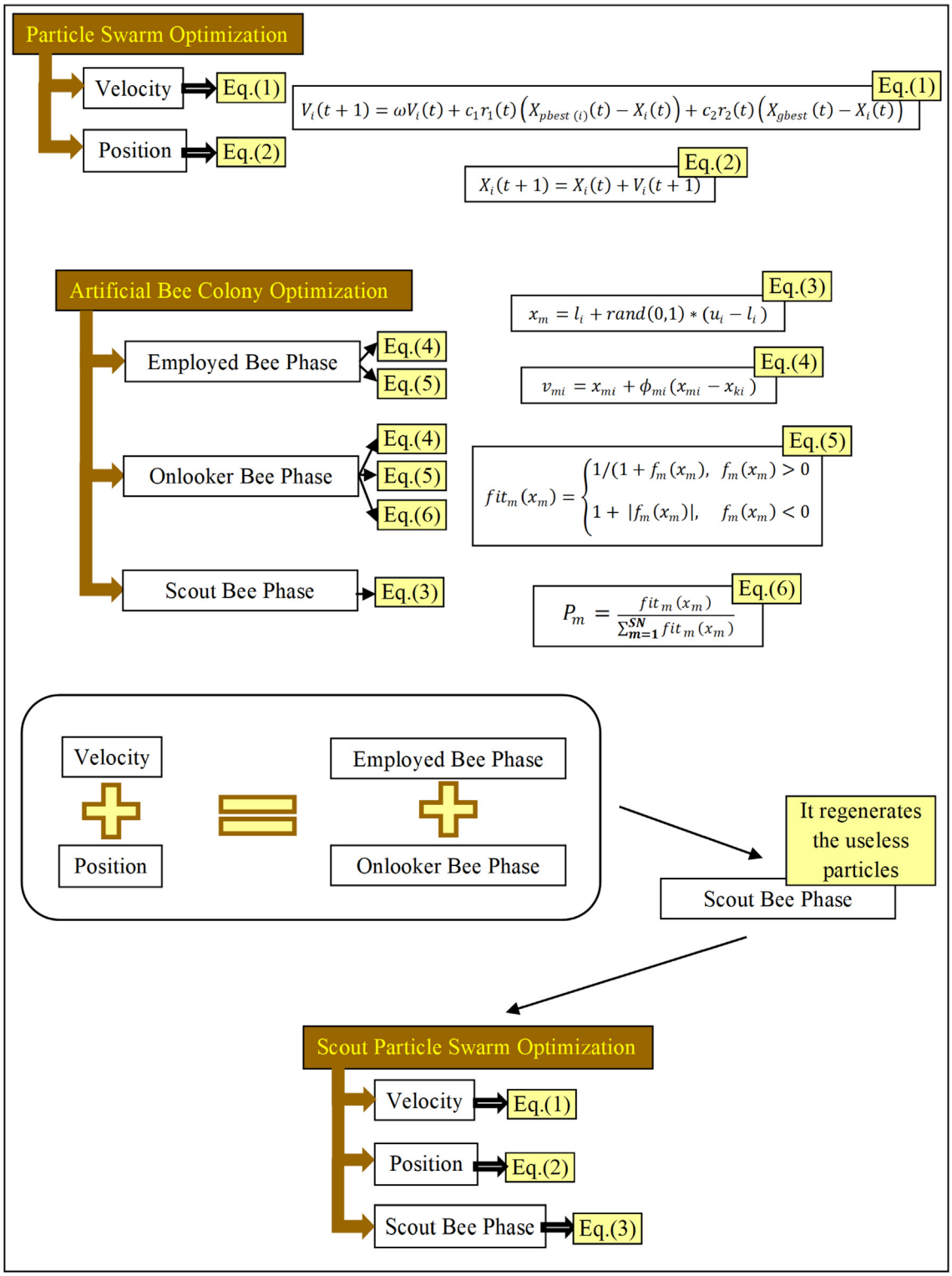

Fig. 3. Inspiration of ScPSO algorithm.

The pseudocode of ScPSO is presented in Table 1 (Koyuncu & Ceylan, 2015).

3. Experimental results

In the literature, a novel optimization algorithm is generally compared with the optimization algorithm from which it originated, and with peer-hybrid methods. For this purpose, ScPSO is first compared with PSO based algorithms that are different variants of PSO. Hence, Global PSO-w, Global PSO-cf, Local PSO-w, Local PSO-cf, UPSO, CLPSO, SG-PSO and SP-PSO techniques (Shin & Kita, 2014) are used for comparison on 8 different benchmark functions.

ScPSO was not designed based on ABC algorithm, but it includes an important part of ABC. For this reason, similar to the study of Jia et al. (2014), comparison process is realized with ABC based tech-

Table 1
Pseudocode of ScPSO algorithm.

Pseudocode of ScPSO
-Initialize all particles within the user defined boundaries
(The first best position (Pbest) values are equal to the position of particles)
-Define a limit value within the range [1, (maximum iteration number-1)]
While (iteration number < maximum iteration number)
-Calculate fitness according to the cost function for all particles
-Update the best position values according to fitness values for all particles
-Choose the best Pbest vector as being Gbest (vector achieved to the minimum cost)
-Calculate new positions according to following equations for all particles
$V_i(t+1) = \omega V_i(t)+c_1r_1(X_{pbest(i)}(t)-X_i(t))+c_2r_2(X_{gbest}(t)-X_i(t))$
$X_i(t+1) = X_i(t)+V_i(t+1)$
-If a variable inertia weight is used, change it in accordance with the utilized rule
-Control all particles which exceed the parameter 'limit', then regenerate the useless ones
End

niques (normABC, BinABC and DisABC) and with GA in the second part of study.

In the third part of study, ScPSO is compared with the efficient – hybrid algorithms (HPA, ABC-PS and PS-ABC) designed by using PSO and ABC algorithms (Li et al., 2015). Thus, the formation of ScPSO is tested whether it constitutes a need for the literature or it does not.

3.1. Problems and experimental settings

Before beginning the comparison process, all ScPSO parameters are examined to achieve optimal operating conditions (optimal values and optimal ranges). The parameter *limit* not only affects the diversity, but it can also change the dynamics of the basic PSO parameters. To determine the best operating conditions, 5 non-linear benchmark functions (Sphere, Rosenbrock, Rastrigin, Griewank and Schaffer's f6) are used as suggested (Shin & Kita, 2014). These well-known functions and their operating conditions are shown in Table 2. The *Threshold* values stand for the targets to be found by optimization algorithms. In other words, the optimization algorithm should reach the *Threshold* value for particle values defined within the boundary of $[X_{min}, X_{max}]$; the length of a particle vector is chosen according to the *Dimension*.

To obtain the best parameter values and ranges, speedy convergence is desired in achieving the threshold values shown in Table 2. Experimental studies are performed using five nonlinear benchmark functions, and the results are evaluated using four statistical metrics that are average iteration number, standard deviation of all iterations, minimum value of all iterations and search rate (the ratio of the successful iteration number to the total iteration number). By this means, parameter values and ranges are compared based on Total Statistical Success (TSS), which is defined as the number of best performances (or the number of obtained first rows).

Table 2
Benchmark functions and operating conditions in trials.

Function	Dimension	$[X_{min}, X_{max}]$	Threshold
$f_{sphere}(\boldsymbol{x}) = \sum_{i=1}^{n} \boldsymbol{x}_i^2$	30	[−100,100]	0.01
$f_{rosenbrock}(\boldsymbol{x}) = \sum_{i=1}^{n-1}\left(100(\boldsymbol{x}_{i+1}-\boldsymbol{x}_i^2)^2+(\boldsymbol{x}_i-1)^2\right)$	30	[−30,30]	100
$f_{rastrigin}(\boldsymbol{x}) = \sum_{i=1}^{n}(\boldsymbol{x}_i^2-10\cos(2\pi\boldsymbol{x}_i)+10)$	30	[−5.12,5.12]	100
$f_{griewank}(\boldsymbol{x}) = \frac{1}{4000}\sum_{i=1}^{n}\boldsymbol{x}_i^2-\prod_{i=1}^{n}\cos\left(\frac{\boldsymbol{x}_i}{\sqrt{i}}\right)+1$	30	[−600,600]	0.1
$f_{schaffer'sf6}(\boldsymbol{x}) = 0.5+\frac{\left(\sin\sqrt{(\boldsymbol{x}_1^2+\boldsymbol{x}_2^2)}\right)^2-0.5}{\left(1+0.001(\boldsymbol{x}_1^2+\boldsymbol{x}_2^2)\right)^2}$	2	[−100,100]	0.00001

Table 3
Performance comparison of different limit values.

Limit		5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100	150	200	250	300
AIN	Griewank	7662	4325	5134	5262	5966	4885	7220	7280	7189	5018	7360	6293	6755	7247	**4019**	7668	5955	7118	6787	6921	6821	7181	7513	6353
	Rastrigin	1534	1294	479	416	950	1206	781	1893	296	980	944	375	1361	391	**273**	536	715	1195	1875	455	457	1102	834	1079
	Sphere	5820	5899	5386	5793	4544	5406	4928	7741	7197	5843	5813	6817	7265	4926	**3496**	6003	5850	5800	6803	6303	5827	6293	6401	5876
	Rosenbrock	3945	3657	4324	4215	4939	5311	5262	5376	4990	5069	3798	4729	4836	5368	**2927**	5368	4204	3622	4521	5361	5175	5259	4514	5731
	Schaffer's f6	2604	2382	2336	3408	2291	2082	2955	2052	1331	1474	4416	2630	1353	1300	2453	**546**	2078	1549	1469	2026	2558	1930	1878	2295
Std	Griewank	**3952**	4383	4444	4632	4471	4630	4248	4172	4295	4525	4060	4543	4423	4208	4398	4040	4480	4109	4382	4237	4309	4307	3996	4473
	Rastrigin	1769	2491	511	292	2122	2341	2133	2920	315	1567	1430	496	1997	305	**274**	689	826	2882	2815	541	599	2104	2177	1558
	Sphere	4623	4539	4615	4652	4482	4596	4590	**3913**	4281	4597	4632	4342	4183	4591	4259	4436	4594	4644	4359	4529	4614	4541	4423	4562
	Rosenbrock	3705	3149	3816	3945	3642	3905	4199	3919	3824	4088	3383	4007	3780	3941	**2806**	3958	3497	3637	3761	4260	4396	4090	3737	3618
	Schaffer's f6	3868	3219	3280	3879	3421	2569	3541	3424	2227	1858	3983	3202	1535	1415	3272	**534**	2400	2328	2876	2283	2843	2777	3057	3216
Min	Griewank	536	485	471	466	500	**385**	487	519	450	466	491	441	466	623	482	529	548	569	518	480	553	455	554	434
	Rastrigin	95	59	48	54	66	64	46	89	**38**	72	99	87	61	61	44	50	58	48	105	44	47	81	72	78
	Sphere	527	572	634	475	484	589	610	712	543	556	**392**	545	532	624	514	545	463	544	668	559	550	543	564	602
	Rosenbrock	778	820	1062	537	1012	993	722	719	1049	569	524	840	797	782	800	568	755	760	814	**488**	616	676	856	987
	Schaffer's f6	36	28	42	61	31	60	73	28	59	50	122	49	53	51	**23**	103	78	25	33	172	69	109	**23**	38
SR	Griewank	0,30	**0,65**	0,55	0,60	0,45	0,55	0,30	0,30	0,30	0,55	0,30	0,40	0,35	0,30	**0,65**	0,25	0,45	0,35	0,35	0,35	0,40	0,30	0,30	0,40
	Rastrigin	**1**	0,95	**1**	**1**	0,95	0,95	0,95	0,90	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0,95	0,90	**1**	**1**	0,95	0,95	**1**
	Sphere	0,45	0,45	0,50	0,45	0,60	0,50	0,55	0,25	0,30	0,45	0,45	0,35	0,30	0,55	**0,70**	0,45	0,45	0,45	0,35	0,40	0,45	0,40	0,40	0,45
	Rosenbrock	0,75	**0,90**	0,70	0,70	0,75	0,60	0,60	0,60	0,65	0,65	0,80	0,70	0,70	0,65	**0,90**	0,65	0,80	0,80	0,75	0,60	0,55	0,60	0,70	0,65
	Schaffer's f6	0,80	0,95	0,95	0,85	0,85	0,95	0,85	0,85	0,95	**1**	0,75	0,90	**1**	**1**	0,90	**1**	0,95	0,95	0,90	**1**	0,95	0,95	0,90	0,90
Total Statistical Success		2	2	1	1	0	1	0	1	2	2	2	1	2	2	**11**	4	1	0	0	3	1	0	1	1

Average Iteration Number = AIN, Standard Deviation = Std, Minimum Value = Min, Success Rate = SR.

The successful iteration in search rate is one at which global convergence is reached. Among four metrics, average iteration number stands for the most important one as in (Jia et al., 2014).

Population size, maximum iteration number and the number of independent runs are respectively 30, 10,000 and 20 as in (Shin & Kita, 2014). Dimensions for Schaffer's f6 and for the other functions are 2 and 30, respectively. Velocity boundaries ($[V_{min}, V_{max}]$) were changed in fractional folds of position boundaries ($[X_{min}, X_{max}]$). As seen in Eq. (7), k represents the fractional fold number.

$$[V_{min}, V_{max}] = [X_{min}, X_{max}] * k \tag{7}$$

However, the coefficients c_1 and c_2 are altered according to the rule that their sum is equal to 4, more than 4 and less than 4. While doing this, their sum is fixed within the range [3.5, 4.5] according to the advice of Zhang (2012). The inertia weight is analyzed within the range [0,1], and linear descent movement (0.9 → 0.4) is added to the trials as suggested (Shi & Eberhart, 1999). The parameter *limit* is examined over a wide range, and a relationship between *limit* and dimension-population size is searched. The comparisons of different parameter values and ranges (based on average iteration number, standard deviation, minimum value and search rate) are shown in Tables 3–6.

As seen in Table 3, the best value for *limit* is 75 at which superior performance is achieved based on the average iteration number, standard deviation and success rate. To observe this, consider the Total Statistical Success (achieved on the first rank). TSS is 11 when *limit* is set at 75. The second TSS value is 4 for a *limit* value of 80. Therefore, it is obvious that selection of *limit* as 75 is more reliable.

Table 4
Performance comparison of different velocity boundaries.

k (fractional fold)		2	1.8	1.6	1.4	1.2	1	0.8	0.6	0.4	0.2
AIN	**Griewank**	5835	4121	5838	7661	4608	**4019**	5792	7202	8061	8610
	Rastrigin	6572	5117	4570	4396	2704	273	221	127	49	**27**
	Sphere	4077	4509	7231	3537	6773	**3496**	5388	6715	8146	8571
	Rosenbrock	4962	5171	3850	5029	4861	**2927**	3319	3949	4814	8178
	Schaffer's f6	2112	2620	2151	2236	2489	**1762**	4046	2733	1989	2169
Std	**Griewank**	4607	4280	4605	4051	4458	4398	4653	4276	3619	**3308**
	Rastrigin	3801	4079	3744	3931	3633	274	195	113	28	**9**
	Sphere	4370	4487	4230	4232	4400	4259	4615	4478	3708	**3401**
	Rosenbrock	4130	3813	3297	4115	3900	**2806**	3441	3780	4270	3651
	Schaffer's f6	3103	3716	3081	2956	3788	2877	4097	3641	**2868**	2993
Min	**Griewank**	551	589	484	488	544	482	413	**397**	497	672
	Rastrigin	478	150	252	456	113	44	54	34	25	**16**
	Sphere	539	568	628	527	510	514	553	457	576	**419**
	Rosenbrock	913	1088	1120	917	818	800	775	700	438	**281**
	Schaffer's f6	43	47	52	66	71	**8**	49	78	51	36
SR	**Griewank**	0,45	**0,70**	0,45	0,25	0,60	0,65	0,45	0,30	0,25	0,15
	Rastrigin	0,50	0,70	0,75	0,70	0,85	**1**	**1**	**1**	**1**	**1**
	Sphere	0,65	0,60	0,30	**0,70**	0,35	**0,70**	0,50	0,35	0,20	0,15
	Rosenbrock	0,60	0,65	0,80	0,60	0,65	**0,90**	0,80	0,75	0,60	0,20
	Schaffer's f6	**0,90**	0,80	**0,90**	**0,90**	0,80	**0,90**	0,75	0,85	**0,90**	**0,90**
Total Statistical Success		1	1	1	2	0	**10**	1	2	3	9

Average Iteration Number = AIN, Standard Deviation = Std, Minimum Value = Min, Success Rate = SR.

Table 5
Performance comparison of different inertia weight values.

Inertia Weight		0.9 → 0.4	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
AIN	**Griewank**	**4019**	6322	5858	6430	7669	6145	6437	6363	5857	6968
	Rastrigin	**273**	1632	793	926	2317	353	1218	1311	914	1180
	Sphere	**3496**	6350	7221	4470	6526	5848	5038	5428	5837	6762
	Rosenbrock	**2927**	6384	4389	5286	4465	3123	3909	5266	4713	6000
	Schaffer's f6	1762	1378	1546	2273	**914**	4025	2400	2000	4207	2475
Std	**Griewank**	4398	4513	4584	4382	**4039**	4324	4395	4461	4582	4188
	Rastrigin	**274**	2438	891	2158	3573	290	2342	2171	980	2326
	Sphere	4259	4475	**4248**	4517	4323	4591	4501	4574	4605	4413
	Rosenbrock	**2806**	4109	3784	4155	3770	3120	3311	3760	4026	4034
	Schaffer's f6	2877	1595	1588	2838	**1419**	4043	3339	3052	3860	3810
Min	**Griewank**	482	**353**	406	650	458	528	477	648	570	608
	Rastrigin	44	90	67	56	55	111	49	57	**33**	62
	Sphere	514	**480**	508	505	521	545	611	483	524	634
	Rosenbrock	800	858	794	799	656	668	**556**	856	850	981
	Schaffer's f6	8	49	31	65	21	37	**6**	60	106	38
SR	**Griewank**	**0,65**	0,40	0,45	0,40	0,25	0,45	0,40	0,40	0,45	0,35
	Rastrigin	**1**	0,95	**1**	0,95	0,85	**1**	0,95	0,95	**1**	0,95
	Sphere	**0,70**	0,40	0,30	0,60	0,40	0,45	0,55	0,50	0,45	0,35
	Rosenbrock	**0,90**	0,45	0,70	0,60	0,80	0,85	0,80	0,75	0,65	0,55
	Schaffer's f6	0,90	**1**	**1**	0,95	**1**	0,80	0,85	0,90	0,75	0,80
Total Statistical Success		**10**	3	3	0	4	1	2	0	2	0

Average Iteration Number = AIN, Standard Deviation = Std, Minimum Value = Min, Success Rate = SR.

Table 6
Performance comparison of different acceleration constants.

Acceleration Constants (c_1/c_2)		1.6/ 2.4	1.65/ 2.35	1.7/ 2.3	1.75/ 2.25	1.8/ 2.2	1.85/ 2.15	1.9/ 2.1	1.92/ 2.08	1.94/ 2.06	1.96/ 2.04	1.98/ 2.02	2/2	2.02/ 1.98	2.04/ 1.96	2.06/ 1.94	2.08/ 1.92	2.1/ 1.9	2.15/ 1.85	2.2/ 1.8	2.25/ 1.75	2.3/ 1.7	2.35/ 1.65	2.4/ 1.6
AIN	**Griewank**	2088	**1108**	2970	2249	2884	2738	2595	2545	2645	3087	7274	4019	7255	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS
	Rastrigin	1573	5093	1379	1587	5603	813	500	843	1375	929	848	**273**	419	955	664	278	344	2043	961	726	1250	1735	1449
	Sphere	1242	1309	1049	1054	948	**790**	2078	2572	2661	3931	1721	3496	4451	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS
	Rosenbrock	5993	5029	3486	5206	4304	4960	3839	4837	3888	3358	5807	**2927**	3209	4912	5906	6718	8467	NCS	NCS	NCS	NCS	NCS	NCS
	Schaffer's f6	1687	2045	**473**	2403	2269	2436	2567	2459	3272	2105	1859	1762	2039	2968	1744	1896	2327	2512	890	1730	2761	2540	3002
Std	**Griewank**	2645	**278**	3526	2980	3615	3631	3706	3729	3692	3998	4176	4398	4202	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS
	Rastrigin	2909	3696	2171	2949	4129	845	595	2111	2417	1011	2121	**274**	497	2131	1003	277	648	3503	1779	1300	2330	3065	2445
	Sphere	173	228	194	178	206	**118**	3330	3716	3675	4454	2788	4259	4534	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS
	Rosenbrock	3556	3412	2917	3505	3397	3607	3339	3917	3577	3387	3905	**2806**	3190	3850	4005	3970	2981	NCS	NCS	NCS	NCS	NCS	NCS
	Schaffer's f6	2582	3158	**673**	3337	2756	3515	3333	3177	4030	3235	2880	2877	2572	3409	2559	2429	3073	3424	945	2598	3510	3708	3332
Min	**Griewank**	894	728	654	606	557	571	529	477	522	479	**470**	482	507	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS
	Rastrigin	104	435	156	110	363	68	58	82	52	56	60	44	60	64	101	39	50	61	43	35	36	**27**	39
	Sphere	907	842	655	737	561	529	515	559	552	**489**	556	514	592	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS
	Rosenbrock	1626	1422	1111	1159	1397	1010	937	**588**	787	717	758	800	692	841	720	1032	1304	NCS	NCS	NCS	NCS	NCS	NCS
	Schaffer's f6	53	60	**4**	34	36	54	27	41	31	62	34	8	33	92	59	47	30	66	83	41	6	70	66
SR	**Griewank**	0,90	**1**	0,80	0,90	0,80	0,85	0,80	0,80	0,80	0,75	0,30	0,65	0,30	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS
	Rastrigin	0,90	0,65	0,95	0,90	0,55	**1**	**1**	0,95	0,95	**1**	0,95	**1**	**1**	0,95	**1**	**1**	**1**	0,85	**1**	**1**	0,95	0,90	0,95
	Sphere	**1**	**1**	**1**	**1**	**1**	**1**	0,85	0,80	0,80	0,65	0,90	0,70	0,60	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS	NCS
	Rosenbrock	0,60	0,70	**0,90**	0,75	0,75	0,75	0,80	0,65	0,75	0,80	0,55	**0,90**	0,85	0,65	0,55	0,45	0,25	NCS	NCS	NCS	NCS	NCS	NCS	NCS
	Schaffer's f6	0,95	0,90	**1**	0,85	0,95	0,85	0,85	0,90	0,80	0,90	0,90	0,90	0,95	0,85	0,95	0,95	0,95	0,90	**1**	**1**	0,85	0,85	0,85
Total Statistical Success		1	4	**6**	1	1	4	1	1	0	2	1	**6**	1	0	1	1	1	0	2	2	0	1	0

Not a convenient solution for convergence and speed = NCS & (Average Iteration Number = AIN, Standard Deviation = Std, Minimum Value = Min, Success Rate = SR).

As seen in Table 4, the parameter k (fractional fold shown in Eq. (7)) is searched in the interval [0,2]. It's seen that the first two values of TSS are 10 and 9 which are obtained when $k = 1$ and $k = 0.2$, respectively. Before the determination of an optimum value for k, if the first two TSS values are close to each other, the importance of the average iteration number and success rate should be taken into consideration. These statistical measures are defined as the most used and important ones in determining convergence performance (Shin & Kita, 2014). The average iteration number (AIN) and success rate (SR) measure convergence performance much better than the standard deviation (Std) and minimum value (Min). If TSS is calculated based on AIN and SR, the TSS when $k = 1$ is 8, while it's 3 when $k = 0.2$. In light of these inferences, an optimal value for k is found to be 1. In other words, the position boundary must be equal to the velocity boundary for better convergence. Additionally, in some applications, $k = 0.2$ may be used as an alternative, if $k = 1$ is not reliable.

As seen in Table 5, the best value of TSS is found as 10, and the inertia weight linearly decreases ($0.9 \rightarrow 0.4$) as advised (Shi & Eberhart, 1999). The second best value of TSS is obtained as 4. In other words, there is no need to compare the linear decrease with other options because the gap between them is sufficiently large.

As suggested (Zhang, 2012), the sum of the c_1 and c_2 coefficients is changed to be equal to 3.5, 4 and 4.5. However, when sum of these coefficients is set equal to 3.5 and 4.5, convergence deteriorated sharply. Owing to this result, the choice of the sum as 4 stands for the unique solution to obtain an optimal value. Therefore, the sum of coefficients is fixed to 4.

As seen in Table 6, the first two values of TSS are 6, for $c_1/c_2 = 2/2$ and $c_1/c_2 = 1.7/2.3$. Due to the same TSSs, we investigated the TSS of AIN and SR for both situations. However, it is seen that this value was the same, 4, for both of them. At this time, we must focus on the metrics Std and Min. In the literature, it is obvious that Std is the more preferred metric since it reflects the determination of convergence. The value of TSS based on Std is 2 and 1 respectively for $c_1/c_2 = 2/2$ and $c_1/c_2 = 1.7/2.3$. Hence it can be inferred that the optimum values for the acceleration constants are $c_1/c_2 = 2/2$. In addition, we suggest that $c_1/c_2 = 1.7/2.3$ be taken into consideration in different applications in which $c_1/c_2 = 2/2$ is not effective.

When the optimal conditions are examined, note that trials not defined in the tables were performed. However, they were of no consequence because of poor convergence performance. In behalf of not moving away from the aim, important parameter changes are presented in tables.

According to the results, the optimal parameter values and ranges are:

- *Limit* = 75
- $[V_{min}, V_{max}] = [X_{min}, X_{max}]$ & $k = 1$
- Inertia weight = $0.9 \rightarrow 0.4$
- $c_1/c_2 = 2/2$

When the optimal conditions are considered, it is seen that ScPSO has approximately the same dynamics as Standard PSO. All the parameters (except *limit*) have the same optimal values as in Standard PSO. However, a few alternative approaches for parameter selection are available. In unbalanced situations (bad convergence), $k = 0.2$ can be the alternative value to be used instead of $k = 1$. In the same way, $c_1/c_2 = 1.7/2.3$ can be an alternative choice if the option of $c_1/c_2 = 2/2$ is not effective with regards to the convergence. These second best conditions (alternative suggestions) should only be used when there is a need. Otherwise, ScPSO is not operated under optimal conditions.

3.2. Performance evaluation of ScPSO

After attaining the optimal conditions, ScPSO is compared with variants of PSO (Shin & Kita, 2014) on 8 numerical benchmark problems (Sphere, Rosenbrock, Schwefel, Weierstrass, Shifted

Table 7
The used benchmark functions on comparisons.

Function	x*	f(x*)	Search Space
$\boldsymbol{f}_1 = \sum_{i=1}^{\boldsymbol{n}} \boldsymbol{x}_i^2$	$[0,0]^n$	0	$[-100,100]$
$\boldsymbol{f}_2 = \sum_{i=1}^{\boldsymbol{n}-1} \left(100(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i^2)^2 + (\boldsymbol{x}_i - 1)^2\right)$	$[1,1]^n$	0	$[-2.048, 2.048]$
$\boldsymbol{f}_3 = 418.9829\boldsymbol{xn} - \sum_{i=1}^{\boldsymbol{n}} \left(\boldsymbol{x}_i \sin\left(\lvert\boldsymbol{x}_i\rvert^{1/2}\right)\right)$	$[420.968,420.968]^n$	0	$[-500,500]$
$\boldsymbol{f}_4 = \sum_{i=1}^{\boldsymbol{n}} \left(\sum_{k=0}^{\boldsymbol{k_{max}}} \left[\boldsymbol{a}^{\boldsymbol{k}} \cos\left(2\pi \boldsymbol{b}^{\boldsymbol{k}}(\boldsymbol{x}_i + 0.5)\right)\right]\right) - \boldsymbol{n}\sum_{k=0}^{\boldsymbol{k_{max}}} \left[\boldsymbol{a}^{\boldsymbol{k}} \cos\left(2\pi \boldsymbol{b}^{\boldsymbol{k}} 0.5\right)\right]$ $\boldsymbol{a} = 0.5, \boldsymbol{b} = 3, \boldsymbol{k_{max}} = 20$	$[0,0]^n$	0	$[-0.5,0.5]$
$\boldsymbol{f}_5 = \sum_{i=1}^{\boldsymbol{n}} \left(\sum_{j=1}^{\boldsymbol{i}} \boldsymbol{x}_j\right)^2$	$[0,0]^n$	0	$[-100,100]$
$\boldsymbol{f}_6 = \sum_{i=1}^{\boldsymbol{n}} [\boldsymbol{x}_i + 0.5]^2$	$[0,0]^n$	0	$[-100,100]$
$\boldsymbol{f}_7 = \sum_{i=1}^{\boldsymbol{n}} \boldsymbol{i}\boldsymbol{x}_i^4 + \boldsymbol{rand}[0.1)$	$[0,0]^n$	0	$[-1.28,1.28]$
$\boldsymbol{f}_8 = \boldsymbol{\max}\lvert\boldsymbol{x}_i\rvert$	$[0,0]^n$	0	$[-100,100]$
$\boldsymbol{f}_9 = \frac{1}{4000}\sum_{i=1}^{\boldsymbol{n}} \boldsymbol{x}_i^2 - \prod_{i=1}^{\boldsymbol{n}} \cos\left(\frac{\boldsymbol{x}_i}{\sqrt{\boldsymbol{i}}}\right) + 1$	$[0,0]^n$	0	$[-600,600]$
$\boldsymbol{f}_{10} = 10^6 \boldsymbol{x}_1^2 + \sum_{i=2}^{\boldsymbol{n}} \boldsymbol{x}_i^2$	$[0,0]^n$	0	$[-100,100]$
$\boldsymbol{f}_{11} = \boldsymbol{x}_1^2 + 10^6 \sum_{i=2}^{\boldsymbol{n}} \boldsymbol{x}_i^2$	$[0,0]^n$	0	$[-100,100]$
$\boldsymbol{f}_{12} = \frac{10}{\boldsymbol{n}^2}\prod_{i=1}^{\boldsymbol{n}} \left(1 + \boldsymbol{i}\sum_{j=1}^{32} \frac{\lvert 2^j \boldsymbol{x}_i - \boldsymbol{round}(2^j \boldsymbol{x}_i)\rvert}{2^j}\right)^{\frac{10}{\boldsymbol{n}^{1.2}}} - \frac{10}{\boldsymbol{n}^2}$	$[0,0]^n$	0	$[-100,100]$
$\boldsymbol{f}_{13} = \lvert\sum_{i=1}^{\boldsymbol{n}} \boldsymbol{x}_i^2 - \boldsymbol{n}\rvert^{1/4} + (0.5\sum_{i=1}^{\boldsymbol{n}} \boldsymbol{x}_i^2 + \sum_{i=1}^{\boldsymbol{n}} \boldsymbol{x}_i)/\boldsymbol{n} + 0.5$	Unknown	Unknown	$[-100,100]$
$\boldsymbol{f}_{14} = \left\lvert(\sum_{i=1}^{\boldsymbol{n}} \boldsymbol{x}_i^2)^2 - (\sum_{i=1}^{\boldsymbol{n}} \boldsymbol{x}_i)^2\right\rvert^{1/2} + (0.5\sum_{i=1}^{\boldsymbol{n}} \boldsymbol{x}_i^2 + \sum_{i=1}^{\boldsymbol{n}} \boldsymbol{x}_i)/\boldsymbol{n} + 0.5$	Unknown	Unknown	$[-100,100]$
$\boldsymbol{f}_{15} = \sum_{i=1}^{\boldsymbol{n}} \lvert\boldsymbol{x}_i\rvert + \prod_{i=1}^{\boldsymbol{n}} \lvert\boldsymbol{x}_i\rvert$	$[0,0]^n$	0	$[-10,10]$
$\boldsymbol{f}_{16} = \sum_{i=1}^{\boldsymbol{n}} \left(\sum_{j=1}^{\boldsymbol{i}} \boldsymbol{z}_j\right)^2 - 450$	$[o,o]^n$	−450	$[-100,100]$
$\boldsymbol{f}_{17} = \sum_{i=1}^{\boldsymbol{n}} (\boldsymbol{z}_i^2 - 10\cos(2\pi \boldsymbol{z}_i) + 10) - 330$	$[o,o]^n$	−330	$[-5,5]$
$\boldsymbol{f}_{18} = \sum_{i=1}^{\boldsymbol{n}} (\boldsymbol{z}_i^2 - 10\cos(2\pi \boldsymbol{z}_i) + 10) - 330$	$[o,o]^n$	−330	$[-5,5]$
$\boldsymbol{f}_{19} = \sum_{i=1}^{\boldsymbol{n}} \left(\sum_{k=0}^{\boldsymbol{k_{max}}} \left[\boldsymbol{a}^{\boldsymbol{k}} \cos\left(2\pi \boldsymbol{b}^{\boldsymbol{k}}(\boldsymbol{z}_i + 0.5)\right)\right]\right) - \boldsymbol{n}\sum_{k=0}^{\boldsymbol{k_{max}}} \left[\boldsymbol{a}^{\boldsymbol{k}} \cos\left(2\pi \boldsymbol{b}^{\boldsymbol{k}} 0.5\right)\right] + 90,$ $\boldsymbol{a} = 0.5, \boldsymbol{b} = 3, \boldsymbol{k_{max}} = 20$	$[o,o]^n$	90	$[-0.5,0.5]$

z is $(x\text{-}o)$ for f_{16} and f_{17}, and $(x\text{-}o) * M$ for f_{18} and f_{19}. $o = [o_1, o_2, \ldots, o_n]$ is the shifted global optimum. M is the nxn orthogonal matrix.

Schwefel's Problem 1.2, Shifted Rastrigin, Shifted Rotated Rastrigin, Shifted Rotated Weierstrass), with variants of ABC (Jia et al., 2014) on 8 benchmark functions (Sphere, Schwefel's Problem 1.2, Rosenbrock, Step, Noisy Quartic Function, Schwefel's Problem 2.26, Rastrigin, Griewank), and with PSO–ABC based hybrid techniques (Li et al., 2015) on 8 benchmark functions (Rosenbrock; Weierstrass, Griewank, Discus, Bent Cigar, Katsuura, HappyCat, HGBat). As seen in Table 7, the functions used are defined as f_1 (Sphere), f_2 (Rosenbrock), f_3 (Schwefel), f_4 (Weierstrass), f_5 (Schwefel's Problem 1.2), f_6 (Step), f_7 (Noisy Quartic Function), f_8 (Schwefel's Problem 2.21), f_9 (Griewank), f_{10} (Discus), f_{11} (Bent Cigar), f_{12} (Katsuura), f_{13} (HappyCat), f_{14} (HGBat), f_{15} (Schwefel's Problem 2.22), f_{16} (Shifted Schwefel), f_{17} (Shifted Rastrigin), f_{18} (Shifted Rotated Rastrigin) and f_{19} (Shifted Rotated Weierstrass).

Features of the used functions are defined in Table 8 (Beyer & Finck, 2012; Bossek, 2017; Geitle, 2017; Jamil & Yang, 2013; Katsuura, 1991; Pan et al., 2013; Suganthan et al., 2005; Yao, Liu, & Lin, 1999).

During the comparison process, both of operating conditions (optimum values and alternative solutions) are considered. In addition, the fitness error of the algorithms is investigated using Eq. (8) (Shin & Kita, 2014).

$$Fitness\ \ Error = |f_i(x) - f_i(x^*)| \quad i = (1, 2, \cdots, n) \tag{8}$$

In Eq. (8), $f_i(x)$ means the result obtained using the optimization method, and $f_i(x^*)$ symbolizes the desired convergence value.

Karaboga and Akay (2009) recommended that the values below 10^{-12} are assumed to be 0. In this study, error results for the functions are accepted as '0' for values below 10^{-30}, and the results below this value are generally meaningless/redundant for real-time (online) systems. Additionally, the results below 10^{-30} do not have sufficient areas of application.

Benchmark functions used for performance analysis are taken from CEC2005, CEC2013, CEC 2014, CEC2015, and from the studies of Suganthan et al. (2005) and Yao et al. (1999). These functions are utilized in a wide range and in different areas to compare optimization algorithms.

Table 8
Features of the utilized benchmark functions.

Benchmark Functions	Features
f_1 (Sphere)	Multimodal, Separable, Scalable, Differentiable
f_2 (Rosenbrock)	Unimodal, Non-Separable, Scalable, Differentiable
f_3 (Schwefel)	Unimodal, Partially-Separable, Scalable, Differentiable
f_4 (Weierstrass)	Multimodal, Separable, Scalable, Differentiable
f_5 (Schwefel's Problem 1.2)	Unimodal, Non-Separable, Scalable, Differentiable
f_6 (Step)	Unimodal, Separable, Scalable, Non-Differentiable
f_7 (Noisy Quartic Function)	Multimodal, Separable, Scalable, Differentiable
f_8 (Schwefel's Problem 2.21)	Unimodal, Separable, Scalable, Non-Differentiable
f_9 (Griewank)	Multimodal, Non-Separable, Scalable, Differentiable
f_{10} (Discus)	Unimodal, Separable
f_{11} (Bent Cigar)	Unimodal, Non-Separable, Scalable, Differentiable
f_{12} (Katsuura)	Multimodal, Non-Separable, Non-Differentiable
f_{13} (HappyCat)	Multimodal, Non-Separable, Scalable
f_{14} (HGBat)	Multimodal, Non-Separable
f_{15} (Schwefel's Problem 2.22)	Unimodal, Non-Separable, Scalable, Differentiable
f_{16} (Shifted Schwefel)	Unimodal, Non-Separable, Scalable, Shifted
f_{17} (Shifted Rastrigin)	Multimodal, Separable, Scalable, Shifted
f_{18} (Shifted Rotated Rastrigin)	Multimodal, Non-Separable, Scalable, Shifted, Rotated
f_{19} (Shifted Rotated Weierstrass)	Multimodal, Non-Separable, Scalable, Shifted, Rotated

3.2.1. Performance evaluation of PSO variants

As suggested, population size, maximum iteration number, number of independent runs and dimension are respectively arranged as 30, 10000, 20 and 30 (Shin & Kita, 2014).

Table 9 shows the performance comparison based on mean fitness values. In making the comparison, the rankings of the algorithms were sorted according to their success based on convergence performance (rank = row of obtained results for all algorithms). A fair and objective comparison was performed. The best convergence results are in bold in Table 9.

According to the results in Table 9, for f_3, f_{16} and f_{17}, ScPSO converges more efficiently than other methods, and the difference between ScPSO and the second best algorithm stays as large. Based on the features of these functions, ScPSO proves its effectiveness and satisfies the needs of unimodal, multimodal, separable, fully non-separable, etc. conditions. Additionally, for other functions, the rank of ScPSO never falls below 3rd place meaning that its rank varies between 1 and 3 (1, 2, or 3).

Total Average Rank (TAR) adds up to the mean of all ranks obtained in all the function trials (TAR = mean (all ranks of an algorithm)). The best TAR (1.625) belongs to the proposed method. Additionally, the rank row of ScPSO is 1st, and the 2nd best algorithm is revealed as SP-PSO proposed by Shin and Kita (2014). TAR of SP-PSO is achieved as 1.875 which is the only result close to the proposed method. In the comparison based on TAR, ScPSO is arisen as the best approach that is one step above SP-PSO.

When a comparison based on TSS was performed, ScPSO achieved the best convergence on five of the eight benchmark functions (Sphere, Schwefel, Shifted Schwefel's Problem 1.2, Shifted Rastrigin and Shifted Rotated Weierstrass), while SP-PSO has only two best convergence rates in the results obtained for the Rosenbrock and Weierstrass functions.

For an in-depth analysis about comparison of PSO based variants, ScPSO cannot achieve the best results on f_2, f_4 and f_{18} at which proposed method obtains a similar result to the best score. ScPSO outperforms to peer techniques in other trials. For different conditions like shifted & rotated, unimodal & non-separable, non-separable & differentiable, unimodal & partially-separable etc., the results of ScPSO seem remarkable on continuous function optimization.

With these results, it can be inferred that ScPSO is superior not only to the old versions of PSO but also to two recent PSOs (SP-PSO and SG-PSO).

3.2.2. Comparison of ScPSO with ABC variants

As suggested, population size, maximum iteration number and independent runs are respectively set to 40, 1000 and 25 (Jia et al., 2014).

In the trials of this section, the dimension of all functions is chosen as 20. In other words, the response of ScPSO is investigated under different circumstances (dimension, population size, maximum iteration number and independent runs) than the previous comparison (stated in Section 3.2.1. By doing this, the consistency of ScPSO will be revealed.

Except for the parameter *limit*, the optimal parameter values, ranges and alternative solutions found in Section 3.1 are fixed. The value of *limit* is set at 67 for this comparison. The reason for the assignment of *limit* to 67 is related to a deduction that is presented in the Discussion section. In Table 10, ScPSO is compared with ABC variants (Jia et al., 2014) and with GA (Jia et al., 2014) by using the mean fitness error for the 8 benchmark functions.

According to Table 10, ScPSO converges much better than the other methods for f_1, f_8 and f_{15}. This is because ScPSO is more reli-

Table 9
Comparison of ScPSO with PSO variants.

Function	Global PSO-w	Global PSO-cf	Local PSO-w	Local PSO-cf	UPSO	CLPSO	SG-PSO	SP-PSO	ScPSO
Sphere (f1)	1.00e+03	1.82e+03	1.79e+02	1.34e+04	9.07e+03	**0.00e+00**	**0.00e+00**	1.48e−06	**0.00e+00**
Rank	5	6	4	8	7	1	1	2	1
Rosenbrock (f2)	1.07e+02	4.60e+02	1.27e+02	1.72e+03	1.03e+03	4.81e+01	2.08e+02	**2.20e+01**	3.10e+01
Rank	4	7	5	9	8	3	6	1	2
Schwefel (f3)	4.46e+03	5.84e+03	6.41e+03	7.36e+03	7.98e+03	5.46e+03	4.20e+03	4.03e+03	**2.21e+03**
Rank	4	6	7	8	9	5	3	2	1
Weierstrass (f4)	1.83e+00	2.70e+01	4.37e+00	2.70e+01	2.64e+01	2.42e−01	1.60e+00	**0.00e+00**	3.30e−01
Rank	5	8	6	8	7	2	4	1	3
Shifted Schwefel's Problem 1.2 (f_{16})	6.00e+06	5.88e+06	5.89e+06	1.01e+07	7.85e+06	5.49e+06	5.70e+06	1.73e+05	**3.97e+04**
Rank	7	5	6	9	8	3	4	2	1
Shifted Rastrigin (f_{17})	1.38e+02	2.41e+02	2.93e+02	4.01e+02	2.48e+02	2.16e+02	1.57e+02	1.29e+02	**3.75e+01**
Rank	3	6	8	9	7	5	4	2	1
Shifted Rotated Rastrigin (f_{18})	2.28e+02	3.96e+02	3.83e+02	5.72e+02	3.90e+02	**1.70e+02**	2.31e+02	1.72e+02	1.74e+02
Rank	4	8	6	9	7	1	5	2	3
Shifted Rotated Weierstrass (f_{19})	2.99e+01	3.53e+01	3.37e+01	3.65e+01	3.98e+01	3.27e+01	2.77e+01	2.89e+01	**2.73e+01**
Rank	4	7	6	8	9	5	2	3	1
Total Average Rank (TAR)	**4.5**	**6.625**	**6**	**8.5**	**7.75**	**3.125**	**3.625**	**1.875**	**1.625**
Rank based on TAR	**5**	**7**	**6**	**9**	**8**	**3**	**4**	**2**	**1**
Total Statistical Success (TSS)	**0**	**0**	**0**	**0**	**0**	**2**	**1**	**2**	**5**

Table 10
Comparison of ScPSO with ABC variants and GA.

Function	GA	normABC	BinABC	DisABC	BitABC	ScPSO
Sphere (f1)	1.16e−02	1.42e+02	1.55e+01	8.54e+03	1.82e−07	**1.12e−10**
Rank	3	5	4	6	2	1
Rosenbrock (f2)	3.02e+01	1.42e+05	1.25e+02	3.81e+06	1.81e+01	**1.65e+01**
Rank	3	5	4	6	2	1
Schwefel's Problem 1.2 (f5)	4.40e+03	7.48e+03	3.98e+03	1.28e+05	7.47e+03	**2.02e+03**
Rank	3	5	2	6	4	1
Step (f6)	5.18e+01	1.67e+02	1.89e+01	8.71e+03	**0.00e+00**	4.57e−13
Rank	4	5	3	6	1	2
Quartic Function i.e. Noise (f7)	1.63e−01	3.22e−01	1.24e−01	2.03e+00	5.91e−02	**3.51e−02**
Rank	4	5	3	6	2	1
Schwefel's Problem 2.21 (f8)	1.63e+01	3.33e+01	1.16e+01	3.33e+01	1.42e+01	**4.79e+00**
Rank	4	5	2	5	3	1
Griewank (f9)	7.39e−02	1.98e+00	1.21e+00	7.31e+01	**1.45e−05**	2.45e−02
Rank	3	5	4	6	1	2
Schwefel's Problem 2.22 (f_{15})	1.09e−02	4.48e−01	1.76e+00	3.55e+01	1.91e−04	**1.15e−11**
Rank	3	4	5	6	2	1
Total Average Rank (TAR)	**3.375**	**4.875**	**3.375**	**5.875**	**2.125**	**1.25**
Rank based on TAR	**3**	**4**	**3**	**5**	**2**	**1**
Total Statistical Success (TSS)	**0**	**0**	**0**	**0**	**2**	**6**

able and effective in continuous, differentiable, scalable, multimodal, unimodal, separable and non-separable situations. In addition, ScPSO is revealed as the only method that does not fall below 2nd place (based on rank).

The best TAR (1.25) belongs to ScPSO, and the rank row of ScPSO is found as 1. The 2nd best algorithm is arisen as BitABC proposed by Jia et al. (2014). The TAR of BitABC is 2.125, a result which is not a close to ScPSO's. Therefore, ScPSO scored better than ABC variants in the comparison based on TAR.

Based on TSS, ScPSO has a satisfactory performance with best convergence on six of the eight benchmark functions (Sphere, Rosenbrock, Schwefel's Problem 1.2, Quartic Function i.e., Noise, Schwefel's Problem 2.21 and Schwefel's Problem 2.22).

For an in-depth analysis about comparison of ABC based variants, ScPSO cannot achieve the best results on f_6 and f_9. For Step function (f_6), ScPSO obtains a similar result to the best score, however its performance stays low for multimodal & non-separable situation (f_9). For other conditions like unimodal & separable, unimodal & non-separable, non-separable & differentiable, etc., the results of ScPSO can be considered as valuable on continuous function optimization.

In the light of results, it's seen that ScPSO is partially based on PSO, but it can outperform to both PSO and ABC variants.

3.2.3. Performance comparison with PSO – ABC based hybrid methods

As suggested, population size, maximum iteration number, search space and independent runs are respectively set to 100, 6000, $[-100,100]^n$ and 20 (Li et al., 2015). In the trials of this section, population size and maximum iteration number is kept high as the dimension of all functions is chosen bigger (60). By doing this, the consistency of ScPSO is examined in high dimensional space.

Table 11
Comparison of ScPSO with hybrid techniques based on PSO and ABC algorithms.

Function	HPA	ABC-PS	PS-ABC	ScPSO
Rosenbrock (f2)	1.8851e+01	**4.8637e−02**	5.8672e+01	5.82e+01
Rank	2	**1**	4	3
Weierstrass (f4)	3.7219e−05	1.7459e−03	**5.4677e−14**	3.56e−13
Rank	3	4	**1**	2
Griewank (f9)	5.1582e−02	1.6354e−08	**0**	**0**
Rank	3	2	**1**	**1**
Discus (f_{10})	**0**	2.4697e−03	**0**	**0**
Rank	**1**	2	**1**	**1**
Bent Cigar (f_{11})	**0**	2.2342e−02	**0**	**0**
Rank	**1**	2	**1**	**1**
Katsuura (f_{12})	**0**	**0**	**0**	**0**
Rank	**1**	**1**	**1**	**1**
HappyCat (f_{13})	1.9535e+00	5.7008e−01	6.7675e−01	**4.78e−01**
Rank	4	2	3	**1**
HGBat (f_{14})	7.6691e+00	5.0492e−01	4.2917e−01	**3.26e−01**
Rank	4	3	2	**1**
Total Average Rank (TAR)	2.375	2.125	1.75	**1.375**
Rank based on TAR	4	3	2	**1**
Total Statistical Success (TSS)	3	2	5	**6**

The optimal parameter values, ranges and alternative solutions in Section 3.1 are fixed, except for the *limit*. The value of *limit* is set at 500 for this comparison concerning the rule stated in Discussion section. In Table 11, ScPSO is compared with the hybrid techniques formed by ABC and PSO algorithms. Performance evaluation is performed by using the mean fitness error for the 8 benchmark functions.

According to Table 11, ScPSO achieves the best results for f_9-f_{14}. This is because ScPSO is effective for different conditions (multimodal, unimodal, separable, etc.), and it converges more reliable than others especially in multimodal & non-separable situations (f_{12}-f_{14}). Besides, the rank of ScPSO never falls below the 3rd place for other functions.

ScPSO owns to the best TAR value (1.375), and the rank row of algorithm is revealed as 1. PS-ABC algorithm, proposed by Li et al. (2015), is found as the 2nd best algorithm according to the TAR values.

Based on TSS, ScPSO algorithm presents a remarkable performance with best convergence on six of the eight benchmark functions (Griewank, Discus, Bent Cigar, Katsuura, HappyCat and HGBat).

For an in-depth analysis about comparison of PSO-ABC based hybrid techniques, ScPSO cannot achieve the best results on f_2 and f_4. For Rosenbrock function (f_2), performance of ScPSO stays low according to HPA and ABC-PS methods. However, ScPSO obtains a remarkable result which is close to the best one for multimodal & separable situation (f_4). On continuous function optimization, the results of ScPSO can be considered as remarkable for many situations like multimodal & non-separable, unimodal & separable, multimodal & non-differentiable, etc.

Based on TSS, TAR and average performance, ScPSO outperforms other techniques for different conditions. Although every algorithm is originated from PSO and ABC methods, the combination of reliable parts plays a key role for better convergence. Herein, it's seen that necessary additions or efficient hybridization is needed to perform the better convergence.

4. Discussions

The dynamics of ScPSO were investigated in Section 3.1. To determine the best parameter values and ranges, three strategies were followed:

- If the TSSs between the options are too close, calculate the TSSs based on the average iteration number and success rate (TSS criteria = TSS (AIN) + TSS(SR))
- If the TSSs based on average iteration number and success rate are near each other, observe the TSSs based on standard deviation (TSS criteria = TSS (Std))
- If the TSSs based on the standard deviation are near to each other, but are not appropriate for the selection process, use TSSs based on minimum value (TSS criteria = TSS (Min)).

Related to the inferences stated above, optimal parameter values and ranges have been robustly obtained. In addition to the optimum conditions, alternative situations ($c_1/c_2 = 1.7/2.3$ & $k = 0.2$) were produced, and these choices were used in the event of failure with the obtained optimum conditions. Therefore, we see the performance of ScPSO can be kept high in every circumstance (in different featured functions). Consequently, the produced three strategies with respect to the literature studies, are effective in the attainment of parameter values, ranges and alternatives.

According to the results in Tables 9–11, ScPSO owns to a superior convergence to GA, to variants of the PSO and ABC algorithms, and to the hybrid techniques based on PSO and ABC algorithms. By this measure, it has robust convergence in unimodal, multimodal, separable, partially-separable, non-separable, continuous, differentiable, non-differentiable, scalable, shifted and rotated cases, which places the efficiency of the algorithm in context.

Total Average Rank (TAR) and Total Statistical Success (TSS) were used to generate the comparison. For the first set of rankings (a comparison based on TSS), the other orders (2nd, 3rd, etc.) ranking the convergence, were used to determine performance based on TAR. So, an accurate and objective comparison was realized.

In Section 3.2, the basic optimization parameters (maximum iteration number, dimension, population size and independent runs) were changed as suggested in (Jia et al., 2014; Kiran & Gunduz, 2013; Shin & Kita, 2014). This indicates that the optimal parameter values, ranges and alternatives are obtained effectively under varying conditions. However, a compatible *limit* needs to be determined to maintain high ScPSO performance, in case the variable circumstances pull convergence performance down to undesired and inadequate levels. Eq. (9) was developed based on two important optimization parameters, dimension and population size:

Table 12
Performance comparison of two limit values.

Function	Limit	
	75	67
Sphere (f1)	3.50e−10	**1.12e−10**
Rosenbrock (f2)	1.67e+01	**1.65e+01**
Schwefel's Problem 1.2 (f5)	2.17e+03	**2.02e+03**
Step (f6)	1.13e−12	**4.57e−13**
Quartic Function i.e. Noise (f7)	**3.15e−02**	3.51e−02
Schwefel's Problem 2.21 (f8)	5.21e+00	**4.79e+00**
Griewank (f9)	3.49e−02	**2.45e−02**
Schwefel's Problem 2.22 (f_{15})	5.74e−11	**1.15e−11**
Total Statistical Success (TSS)	**1**	**7**

$$Limit = (dimension * population\,size)/12 \quad (9)$$

In the trials discussed in Section 3.2.1, population size and dimension are both 30. Multiplication of these parameters equals 900. At this point, *limit* has a value of 75 which was obtained using to Eq. (9). In Section 3.2.2 and Section 3.2.3, *limit* value was fixed at 67 and at 500 since it is the optimal value for parameter. In Table 12, the *limit* value of 67 is compared with 75 in order to prove the relationship between *limit* and dimension-population size. Herein, this comparison has been realized in Section 3.2.2, and presents an example to reveal the best *limit* choice whether it's a stable parameter as 75 found in Section 3.2.1 or a variable parameter according to Eq. (9).

As seen in Table 12, TSS is found as 7 when *limit = 67*, while it's only 1 for *limit = 75*. This shows the parameter *limit* in ScPSO is not a constant, since it's a variable parameter that must be adjusted according to Eq. (9).

5. Conclusions

In standard PSO, it is necessary to reproduce inadequate particles. A study eliminating this handicap was performed by Koyuncu and Ceylan (Koyuncu & Ceylan, 2015).

In this paper, a convergence analysis of the algorithm ScPSO has been carried out. In addition, its operating conditions have been investigated, and ScPSO was compared with GA, with variants of PSO and ABC algorithms, and with the hybrid techniques combining the operation of PSO and ABC algorithms. ScPSO exhibits convergence and robustness to variable conditions such as unimodal, multimodal, non-separable, etc. operating systems. A different three step statistical test is recommended for the attainment of optimum operating conditions.

It was seen that with the addition of parameter *limit*, convergence performance can be increased. However, the addition of *limit* should be determined by the operating circumstances, and Eq. (9) should be considered to achieve better performance.

It was revealed that hybridization of PSO and scout bee phase creates a more robust optimization algorithm than GA, PSO, ABC, variants of these algorithms, and hybrid methods designed with the use of PSO and ABC methods. Herein, hybridization has been performed by handling the handicap in PSO, and logically, this process has provided a faster approach. As a result, benchmark function optimization is realized more satisfying by using an efficient algorithm (PSO) and by upgrading its performance with scout bee phase. On most of the benchmark functions, ScPSO represented the most reliable performance by means of better optimal solutions in searching space. On the other hand, the different trials based on various runs, iteration number and population size conceived that proposed method can obtain remarkable performance in different conditions.

In future work, an efficient hybrid optimization algorithm will be developed. PSO, ABC, GA, Artificial Immune Recognition algorithm (AIRS) and Ant Colony Optimization techniques will be used to determine the most robust optimization algorithm for global function optimization.

Conflict of interest

The authors declare that they have no conflict of interest.

Acknowledgement

This work is supported by the Coordinatorship of Konya Technical University's Scientific Research Projects.

References

Beyer, H. G., Finck, S. (2012). HappyCat–A simple function class where well-known direct search algorithms do fail. In: 2012 International conference on parallel problem solving from nature; 2012 Sep 1-5 (pp. 367–376).
Bossek, J. (2017). Smoof: Single-and multi-objective optimization test functions. *R Journal, 9*(1), 103–113.
Ceylan, R., & Koyuncu, H. (2016). A new breakpoint in hybrid particle swarm-neural network architecture: individual boundary adjustment. *International Journal of Information Technology & Decision Making, 15*(06), 1313–1343.
Ceylan, R., & Koyuncu, H. (2018). ScPSO-based multithresholding modalities for suspicious region detection on mammograms. soft computing based. *Medical Image Analysis*, 109–135 (Chapter 7).
Chun-Feng, W., Kui, L., & Pei-Ping, S. (2014). Hybrid artificial bee colony algorithm and particle swarm search for global optimization. *Mathematical Problems in Engineering*.
Clerc, M., & Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation, 6*(1), 58–73.
Eberhart, R. C., & Shi, Y. (200). Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of 2000 congress on evolutionary computation; 2000 Jul 16-19 (pp. 84–88). LA Jolla, CA.
El-Abd, M. (2011). A hybrid ABC-SPSO algorithm for continuous function optimization. In: 2011 IEEE symposium on swarm intelligence (SIS); 2011 Apr 11-15 (pp. 1–6).
Geitle, M. (2017). Improving differential evolution using inductive programming. Master's thesis.
Jamil, M., & Yang, X. S. (2013). A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation, 4*(2), 150–194.
Jia, D., Duan, X., & Khan, M. K. (2014). Binary Artificial Bee Colony optimization using bitwise operation. *Computers & Industrial Engineering, 76*, 360–365.
Karaboga, D. (2005). *An idea based on honey bee swarm for numerical optimization. Technical report-tr06*. Erciyes University, TR.
Karaboga, D., & Akay, B. (2009). A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation, 214*(1), 108–132.
Karaboga, D., & Gorkemli, B. (2014). A quick artificial bee colony (qABC) algorithm and its performance on optimization problems. *Applied Soft Computing, 23*, 227–238.
Kashan, M. H., Nahavandi, N., & Kashan, A. H. (2012). DisABC: A new artificial bee colony algorithm for binary optimization. *Applied Soft Computing, 12*(1), 342–352.
Katsuura, H. (1991). Continuous nowhere-differentiable functions—an application of contraction mappings. *The American Mathematical Monthly, 98*(5), 411–416.
Kennedy, J., & Eberhart, R. (1995). particle swarm optimization. In: Proceedings of IEEE international conference on neural networks; 1995 Nov 27 – Dec 1 (pp. 1942–1948). Perth, WA.
Kennedy, J., & Mendes, R. (2002). Population structure and particle swarm performance. In: Proceedings of 2002 Congress on Evolutionary Computation (CEC '02); 2002 May 12-17 (pp. 1671–1676). Honolulu, HI.
Kiran, M. S., & Gunduz, M. (2013). A recombination-based hybridization of particle swarm optimization and artificial bee colony algorithm for continuous optimization problems. *Applied Soft Computing, 13*(4), 2188–2203.
Koyuncu, H., & Ceylan, R. (2013). Artificial neural network based on rotation forest for biomedical pattern classification. In: 2013 36th International conference on telecommunications and signal processing (TSP); 2013 Jul 2-4 (pp. 581–585).
Koyuncu, H., & Ceylan, R. (2015). Scout particle swarm optimization. In: 6th European conference of the international federation for medical and biological engineering; 2015 Sep 7-11 (pp. 82–85).
Li, Z., Wang, W., Yan, Y., & Li, Z. (2015). PS–ABC: A hybrid algorithm based on particle swarm and artificial bee colony for high-dimensional optimization problems. *Expert Systems with Applications, 42*(22), 8881–8895.
Liang, J. J., Qin, A. K., Suganthan, P. N., & Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE transactions on evolutionary computation, 10*(3), 281–295.
Lukasik, S., & Kowalski, P. (2014). Fully informed swarm optimization algorithms: Basic concepts, variants and experimental evaluation. In: 2014 Federated

conference on computer science and information systems (FedCSIS); 2014 Sep 7-10 (pp. 155–161).
Mendes, R., Kennedy, J., & Neves, J. (2004). The fully informed particle swarm: Simpler, maybe better. *IEEE Transactions on Evolutionary Computation, 8*(3), 204–210.
Pampara, G., & Engelbrecht, A.P. (2011). Binary artificial bee colony optimization. In: 2011 IEEE symposium on swarm intelligence (SIS); 2011 Apr 11-15 (pp. 1–8). Paris, FR.
Pan, J. S., Krömer, P., & Snášel, V. (Eds.). (2013). Genetic and evolutionary computing: Proceedings of the seventh international conference on genetic and evolutionary computing. ICGEC 2013.
Parsopoulos, K. E. (2004). UPSO: A unified particle swarm optimization scheme. *Lecture Series on Computer and Computational Science, 1*, 868–873.
Shi, Y., & Eberhart, R. C. (1999). Empirical study of particle swarm optimization. In: Proceedings of the 1999 congress on evolutionary computation (CEC' 99); 1999 Jul 6-9 (pp. 1945–1950). Washington, DC.
Shin, Y. B., & Kita, E. (2014). Search performance improvement of Particle Swarm Optimization by second best particle information. *Applied Mathematics and Computation, 246*, 346–354.
Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y. P., Auger, A., & Tiwari, S. (2005). Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. KanGAL Report. 2005; 2005005.
Wang, K. K., Lv, Q., Zhao, H. Q., & Zhang, W. (2012). Hybrid algorithm for solving complex constrained optimization problems based on PSO and ABC. *Systems Engineering and Electronics, 34*(6), 1193–1199.
Yang, W., & Pei, Z. (2013). Hybrid ABC/PSO to solve travelling salesman problem. *International Journal of Computing Science and Mathematics, 4*(3), 214–221.
Yao, X., Liu, Y., & Lin, G. (1999). Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation, 3*(2), 82–102.
Zhan, Z. H., Zhang, J., Li, Y., & Shi, Y. H. (2011). Orthogonal learning particle swarm optimization. *IEEE Transactions on Evolutionary Computation, 15*(6), 832–847.
Zhang, J. (2012). Experimental parameter investigations on particle swarm optimization acceleration coefficients. *International Journal of Advancements in Computing Technology, 4*(5), 99–105.
Zhu, G., & Kwong, S. (2010). Gbest-guided artificial bee colony algorithm for numerical function optimization. *Applied Mathematics and Computation, 217*(7), 3166–3173.