American Journal of Mathematical and Computer Applications

Research Article

Volume 1 | Issue 1

Acceleration Particle Swarm Optimization

Satish Gajawada

IIT Roorkee Alumnus

***Corresponding author:** **Satish Gajawada, IIT Roorkee Alumnus, India**

Submitted: 03-Mar-2025 **Accepted:** 07-Mar-2025 **Published:** 11-Mar-2025

***Citation:** Gajawada, S. (2025). Acceleration Particle Swarm Optimization. American Journal of Mathematical and Computer Applications. 1(1), 01-13.*

Abstarct

Particle Swarm Optimization (PSO) is a popular optimization algorithm for solving complex optimization problems. Many PSO algorithms were proposed in literature where Velocity was calculated first and then it was added to position to obtain new position. In this work, a novel algorithm titled "Acceleration Particle Swarm Optimization (AccPSO)" is proposed where acceleration is calculated first and then displacement is obtained next with initial velocity, acceleration and time. Displacement is added to position to get new position. Unlike many PSO algorithms in literature, where iterations and time are used interchangeably, the time "t" in AccPSO algorithm is a continuous variable. In this work, AccPSO, PSO, Acceleration-based Particle Swarm Optimization (APSO) and APSOc (APSO with clamping) are tested on seven benchmark functions. Results obtained are discussed. It has been found that AccPSO with time "t" = 0.1 and "t" = 0.25 between iterations yielded optimal results when tested on benchmark functions.

Keywords: Acceleration, Particle Swarm Optimization, PSO, Acceleration Particle Swarm Optimization, AccPSO.

1. Introduction

In [1], Acceleration based Particle Swarm Optimization (APSO) was proposed. In APSO, a new strategy was employed for updating acceleration coefficients. The work Acceleration Particle Swarm Optimization (AccPSO) in this article is different from APSO proposed in [1].

Acceleration-based Particle Swarm Optimization (APSO) was proposed in [2]. The concept of calculation of acceleration in this article is inspired from the behavior of birds. After calculating acceleration, velocity and position are updated. AccPSO proposed in this work calculates acceleration inspired from the behavior of birds in a similar way as done in [2].

In [3], Acceleration based Particle Swarm Optimization (APSO) was used. This algorithm is based on updating acceleration coefficients in a new way.

Centripetal Accelerated Particle Swarm Optimization (CAPSO) was proposed in [4]. The calculation of acceleration in AccPSO and CAPSO are similar. Both papers calculate acceleration inspired from the behavior of birds. However, velocity and position updating equations are different in CAPSO when compared to AccPSO proposed in this paper.

In [5], Acceleration-Aided Particle Swarm Optimization (A-APSO), was proposed. The calculation of acceleration in A-APSO and AccPSO are same. However, A-APSO is based on iterations and time t equals 1. In AccPSO, there is concept of time introduced between iterations and time t in AccPSO proposed in this work is a continuous variable.

Improved Centripetal Accelerated Particle Swarm Optimization was proposed in [6] where the work in [4] is improved.

In [7], Accelerated Particle Swarm Optimization (APSO) was proposed. An acceleration factor 'a' is multiplied in the velocity update equation. AccPSO proposed in this paper and APSO proposed in [7] are different from each other.

Particle Acceleration-based Particle Swarm Optimization (PA-PSO) was proposed in [8]. In PA-PSO, acceleration is obtained by difference of final and initial velocities whereas in AccPSO, acceleration is inspired by behavior of birds.

In [9], Hybrid Strategy Particle Swarm Optimization (HS-PSO) was proposed. HS-PSO is based on Hook-Jeeves strategy, Cauchy particle mutation mechanism, adaptive weight adjustment and fusion of reverse learning strategy.

A Particle Swarm Optimization algorithm based on Adaptive strategy and Velocity pausing (VASPSO) was proposed in [10]. This algorithm is based on Terminal Replacement Mechanism, Adaptive Strategy, Symmetric Cooperative Swarms, Time-varying inertia weight and velocity pausing.

In [11], Improved Particle Swarm Optimization (IPSO) was proposed. Time varying inertia weight, chaos-based initialization scheme, replacement of inactive particles and Adaptive mutation strategy were used in IPSO algorithm.

A Hybrid Particle Swarm Optimization titled "NDWPSO" was proposed in [12]. This method is based on a mutation strategy from Differential Evolution, spiral shrinkage search strategy, dynamic inertial weight, a new jump out strategy and elite opposition-based learning scheme is used for initialization.

In [13], Particle Swarm Optimization algorithm titled "IPSO" was proposed. IPSO is based on random acceleration coefficient and adaptive decreasing inertia weight.

A new velocity update equation was proposed in [14]. There are 3 components in velocity update equation. Moving towards local best is first component and moving towards global best is second component. Third component is based on moving towards the local best murmuration particle "Mbest". K-means clustering is applied and particles are divided into groups. The best fitness particle in the group to which particle belongs to is taken as "Mbest". Generally, there are only two components in velocity update equation of Particle Swarm Optimization (PSO). In this work an extra third component which is moving towards the local best murmuration particle was introduced.

In [15], improved Particle Swarm Optimization algorithm titled "SCMPSO" was proposed. SCMPSO is based on second order-oscillatory particles which improves PSO algorithm. Recent research and development in Particle Swarm Optimization field can be found in articles [16] to [28].

Section 2 shows Particle Swarm Optimization. Details related to proposed Acceleration Particle Swarm Optimization (AccPSO) algorithm can be found in 3rd Section. Section 4 gives Results and Conclusions are made in Section 5.

2. Particle Swarm Optimization

This section explains Particle Swarm Optimization (PSO) algorithm. Equation Eq1 shows velocity update and Equation Eq2 shows position update.

V(i,k) – Velocity of particle "i" and dimension "k"

w – Intertia weight

c1 - Cognitive acceleration coefficient

r1 – Random number uniformly distributed between 0 and 1

pbest(i,k) – ith Particle best position and dimension "k"

gbest(k) – kth dimension of best position of entire swarm

c2 – Social acceleration coefficient

r2 – Random number uniformly distributed between 0 and

1

Position(i,k) – Current position of ith particle and dimension "k"

V(i,k) = (w * V(i,k)) + (c1 * r1 * (pbest(i,k) – Position(i,k))) + (c2 * r2 * (gbest(k) – Position(i,k))) – (Eq1)

Position(i,k) = Position(i,k) + V(i,k) – (Eq2)

Figure 1: Gives pseudocode of PSO algorithm. Figure. 2 shows flowchart of PSO algorithm.

In line 1, population is initialized. Line 2 shows loop for iterations and line 3 shows loop for each particle. Pbest and gbest are updated in lines 4 and 5. Line 6 loops for each dimension. Velocity is updated in lines 7-9. Position is updated in lines 10-12. Loops are ended in lines 13-15.

1. **Population is initialized**
2. **for** iterations = 1 to maximum iterations
3. **for** i = 1 to Size of population
4. **if** fitness(Position(i)) < fitness(pbest(i)) **then** pbest(i) = Position(i)
5. **if** fitness(pbest(i)) < fitness(gbest) **then** gbest = pbest(i)
6. **for** k = 1 to dimensions
7. Update Velocity using (Eq1)
8. **if** V(i,k) > Vmax **then** V(i,k) = Vmax
9. **if** V(i,k) < Vmin **then** V(i,k) = Vmin
10. Update Position using (Eq2)
11. **if** Position(i,k) > xmax **then** Position(i,k) = xmax
12. **if** Position(i,k) < xmin **then** Position(i,k) = xmin
13. **end for** (k)
14. **end for** (i)
15. **end for** (iterations)

Figure 1: Pseudocode of PSO algorithm

Figure. 2 Flowchart of PSO algorithm

3. Acceleration Particle Swarm Optimization

This section explains proposed Acceleration Particle Swarm Optimization (AccPSO) algorithm. Equation Eq3 shows Cognitive Acceleration. Equation Eq4 shows Social Acceleration. Acceleration is obtained by adding Cognitive Acceleration and Social Acceleration in Equation Eq5. Equation Eq6 gives equation for Displacement. Equation Eq7 gives Position update. In this equation Displacement is added to Position to obtain new Position. Equation Eq8 gives equation for Velocity update. This new Velocity is used as initial Velocity while calculating Displacement in the next iteration.

V(i,k) – Velocity of particle "i" and dimension "k"

c1 - Cognitive acceleration coefficient

r1 – Random number uniformly distributed between 0 and 1

pbest(i,k) – ith Particle best position and dimension "k"

gbest – best position of entire swarm

c2 – Social acceleration coefficient

r2 – Random number uniformly distributed between 0 and 1

Cognitive_Acceleration(i,k) – Cognitive acceleration of ith particle and dimension "k"

Social_Acceleration(i,k) – Social acceleration of ith particle and dimension "k"

Acceleration(i,k) - Acceleration of ith particle and dimension "k"

Displacement(i,k) – Displacement of ith particle and dimension "k"

Position(i,k) – Current position of ith particle and dimension "k"

Cognitive_Acceleration(i,k) = (c1 * r1 * (pbest(i,k) – Position(i,k))) – (Eq3)

Social_Acceleration(i,k) = (c2 * r2 * (gbest(k) – Position(i,k))) – (Eq4)

Acceleration(i,k) = Cognitive_Acceleration(i,k) + Social_Acceleration(i,k) – (Eq5)

Displacement(i,k) = V(i,k)*t + 0.5* Acceleration(i,k)*t*t – (Eq6)

Position(i,k) += Displacement(i,k) – (Eq7)

V(i,k) = V(i,k) + (Acceleration(i,k))*t – (Eq8)

Figure. 3 shows pseudocode of proposed Acceleration Particle Swarm Optimization (AccPSO). Figure. 4 gives flowchart of AccPSO algorithm.

In line 1, population is initialized. Line 2 shows loop for iterations and line 3 shows loop for each particle. Pbest and gbest are updated in lines 4 and 5. Line 6 loops for each dimension. Acceleration, Displacement, Position are calculated in lines 7-11 respectively. Velocity is calculated in lines 12-14. Loops are ended in lines 15-17.

1. **Population is initialized**
2. **for** iterations = 1 to maximum iterations
3. **for** i = 1 to Size of population
4. **if** fitness(Position(i)) < fitness(pbest(i)) **then** pbest(i) = Position(i)
5. **if** fitness(pbest(i)) < fitness(gbest) **then** gbest = pbest(i)
6. **for** k = 1 to dimensions
7. Calculate Acceleration using (Eq5)
8. Calculate Displacament using (Eq6)
9. Calculate Position using (Eq7)
10. **if** Position(i,k) > xmax **then** Position(i,k) = xmax
11. **if** Position(i,k) < xmin **then** Position(i,k) = xmin
12. Calculate Velocity using (Eq8)
13. **if** V(i,k) > Vmax **then** V(i,k) = Vmax
14. **if** V(i,k) < Vmin **then** V(i,k) = Vmin
15. **end for** (k)
16. **end for** (i)
17. **end for** (iterations)

Figure 3: Pseudocode of proposed Acceleration Particle Swarm Optimization (AccPSO)

Figure 4: Flowchart of proposed Acceleration Particle Swarm Optimization (AccPSO)

4. Results

In this work APSO proposed in [2] is slightly modified with velocity and position clamping to obtain APSOc. Algorithms PSO, proposed AccPSO (with Time values "t" = 0.05, 0.1, 0.25, 0.5, 1, 5), Acceleration-based Particle Swarm Optimization (APSO) proposed in [2] and APSOc (APSO with clamping) are applied on Rastrigin, Sphere, Ackley, Rosenbrock, Beale, Booth and Three-Hump Camel benchmark functions.

Table. 1 to Table. 7 shows Optimal Position, Optimal Fitness and Rank for Rastrigin, Sphere, Ackley, Rosenbrock, Beale, Booth and Three-Hump Camel benchmark functions respectively.

Table. 8 shows Ranking of algorithms on benchmark functions.

Figure. 5 to Figure. 11 shows Convergence curves for Rastrigin, Sphere, Ackley, Rosenbrock, Beale, Booth and Three-Hump Camel benchmark functions respectively.

Table. 1 Optimal Position, Optimal Fitness and Rank for Rastrigin Function.

Algorithm	Time (t)	Optimal Position	Optimal Fitness	Rank
PSO	-	['0.995065', '-1.003172', '0.948100']	3.429206	2
AccPSO	0.05	['-0.003548', '-1.080543', '-0.928857']	4.268807	4
AccPSO	0.1	['-0.135719', '0.030434', '-0.028947']	3.788229	3
AccPSO	0.25	['0.917834', '-1.013812', '-0.112928']	5.637378	5
AccPSO	0.5	['1.957149', '-1.064489', '-0.920311']	8.208100	6
AccPSO	1	['1.232075', '-1.028791', '1.840819']	19.602662	7
AccPSO	5	['-2.963075', '-0.926583', '0.248887']	20.943561	8
APSOc	-	['0.000000', '0.000000', '0.000000']	0.000000	1
APSO	-	['-2.963075', '-0.926583', '0.248887']	20.943561	8

From Table. 1 Rank column it can be observed that APSOc performed best followed by PSO in second position and proposed AccPSO (t = 0.1) in third position. APSO performed worst

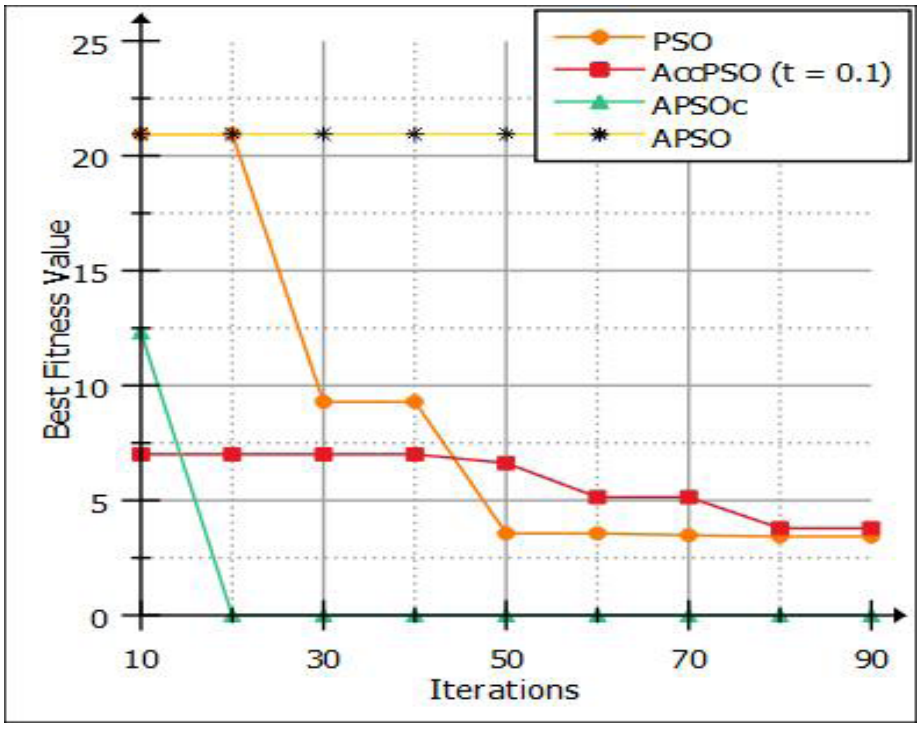

Figure 5: Convergence curve for algorithms PSO, AccPSO (t = 0.1), APSOc, APSO for Rastrigin function

Algorithm	Time (t)	Optimal Position	Optimal Fitness	Rank
PSO	-	['-0.006096', '0.004520', '0.178469']	0.031909	3
AccPSO	0.05	['0.141275', '-0.147410', '0.013723']	0.041877	5
AccPSO	0.1	['0.011334', '0.001608', '0.040052']	0.001735	2
AccPSO	0.25	['-0.035585', '-0.184961', '0.014747']	0.035694	4
AccPSO	0.5	['-0.600960', '-0.660987', '0.612041']	1.172651	8
AccPSO	1	['0.076059', '-0.790917', '0.170985']	0.660570	7
AccPSO	5	['-2.963075', '-0.926583', '0.248887']	9.700313	9
APSOc	-	['0.000000', '0.000000', '0.000000']	0.000000	1
APSO	-	['-0.311292', '-0.430060', '-0.577341']	0.615177	6

Table 2: Optimal Position, Optimal Fitness and Rank for Sphere Function

From Table. 2 Rank column it can be observed that APSOc performed best followed by AccPSO (t = 0.1) in second position and PSO in third position. APSO obtained Rank 6. APSO with clamping (APSOc) obtained Rank 1.

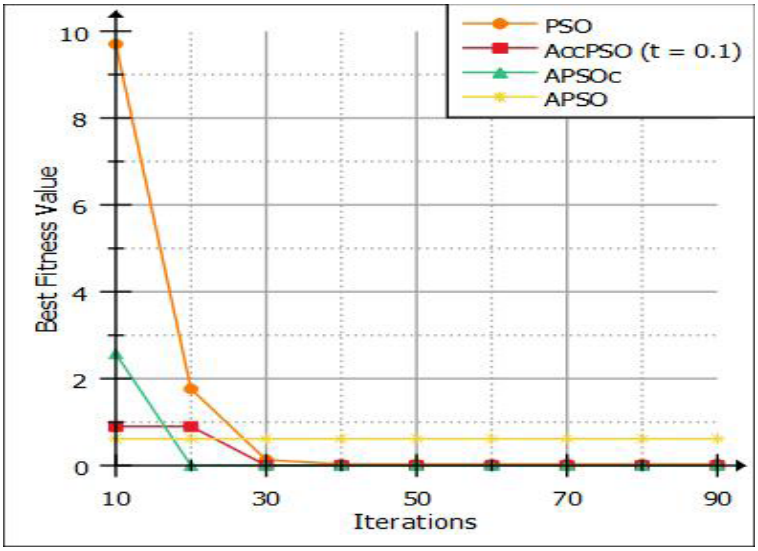

Figure 6: Convergence curve for algorithms PSO, AccPSO (t = 0.1), APSOc, APSO for Sphere function.

Algorithm	Time (t)	Optimal Position	Optimal Fitness	Rank
PSO	-	['0.846509', '0.961760', '0.971649']	3.786924	5
AccPSO	0.05	['-0.010270', '0.177051', '0.572538']	2.857466	3
AccPSO	0.1	['-0.028851', '0.027244', '-0.114688']	0.520206	2
AccPSO	0.25	['-0.379228', '-0.421404', '0.221155']	3.453682	4
AccPSO	0.5	['0.087809', '1.583613', '1.110090']	5.437795	7
AccPSO	1	['-0.684474', '1.295402', '-0.204072']	4.982607	6
AccPSO	5	['-9.709404', '-3.036226', '0.815554']	15.109543	9
APSOc	-	['0.000000', '0.000000', '0.000000']	0.000000	1
APSO	-	['-1.020040', '-1.409219', '-1.891832']	6.491372	8

Table 3: Optimal Position, Optimal Fitness and Rank for Ackley Function

From Table. 3 Rank column it can be observed that APSOc performed best followed by AccPSO (t = 0.1) in second position and PSO in fifth position. APSO obtained Rank 8. APSO with clamping (APSOc) obtained Rank 1

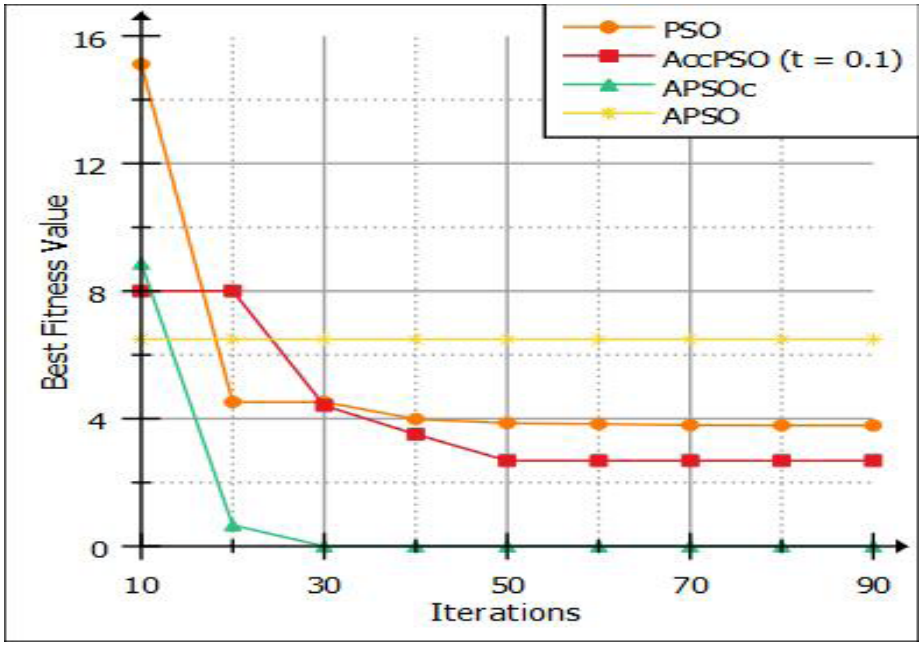

Figure 7: Convergence curve for algorithms PSO, AccPSO (t = 0.1), APSOc, APSO for Ackley function

Algorithm	Time (t)	Optimal Position	Optimal Fitness	Rank
PSO	-	['1.032677', '1.066918', '1.138859']	0.005600	1
AccPSO	0.05	['1.352583', '1.789617', '3.225790']	0.959894	3
AccPSO	0.1	['1.444045', '2.031175', '4.119426']	1.556993	4
AccPSO	0.25	['1.156006', '1.384838', '1.987552']	0.894415	2
AccPSO	0.5	['0.798173', '0.417819', '0.190061']	5.211164	7
AccPSO	1	['-0.500436', '0.231045', '0.195882']	4.910852	6
AccPSO	5	['0.422842', '1.254289', '1.142736']	134.600124	9
APSOc	-	['0.019321', '0.000000', '0.000000']	1.961745	5
APSO	-	['0.172923', '0.656629', '0.578415']	42.248968	8

Table 4: Optimal Position, Optimal Fitness and Rank for Rosenbrock Function

From Table. 4 Rank column it can be observed that PSO performed best followed by AccPSO (t = 0.25) in second position and APSOc in fifth position. APSO obtained Rank 8. APSO with clamping (APSOc) obtained Rank 5.

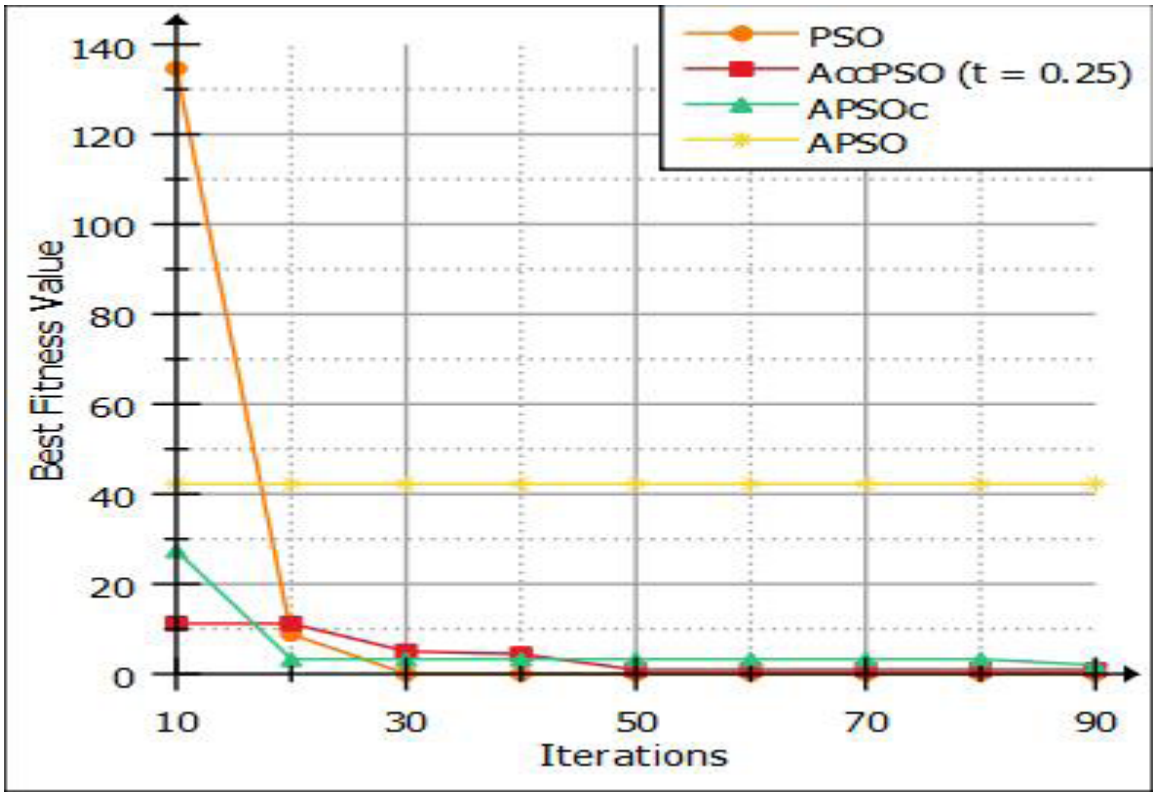

Figure 8: Convergence curve for algorithms PSO, AccPSO (t = 0.25), APSOc, APSO for Rosenbrock function

Algorithm	Time (t)	Optimal Position	Optimal Fitness	Rank
PSO	-	['-3.095066', '1.252973']	0.885911	8
AccPSO	0.05	['2.956021', '0.499396']	0.002761	2
AccPSO	0.1	['3.025866', '0.502864']	0.000392	1
AccPSO	0.25	['3.177125', '0.533403']	0.005739	4
AccPSO	0.5	['3.185840', '0.547782']	0.005305	3
AccPSO	1	['2.841515', '0.480963']	0.014888	5
AccPSO	5	['4.500000', '0.758336']	0.291944	7
APSOc	-	['3.868566', '0.643473']	0.060165	6
APSO	-	['-49.295690', '1.015259']	1.209615	9

Table 5: Optimal Position, Optimal Fitness and Rank for Beale Function

From Table. 5 Rank column it can be observed that AccPSO (t = 0.1) performed best. APSOc obtained sixth position. PSO obtained Rank 8. APSO performed worst.

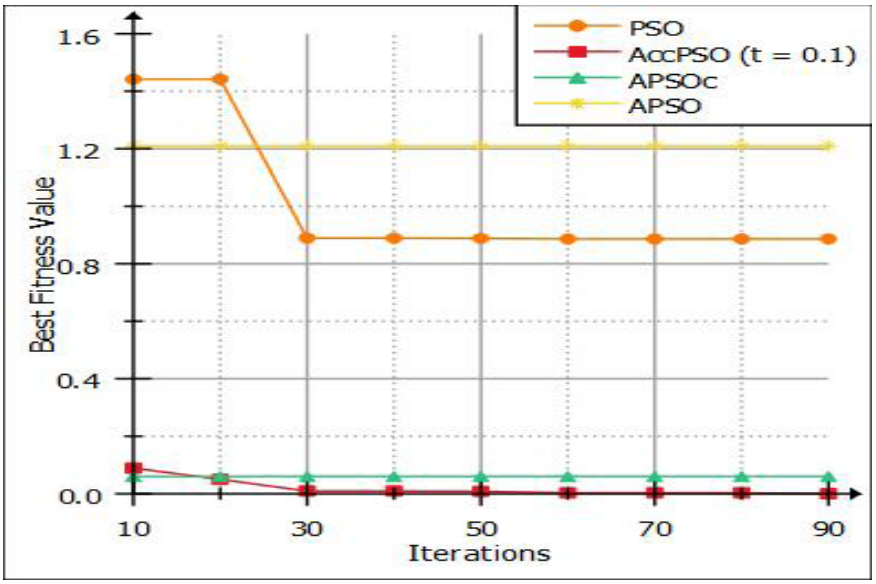

Figure 9: Convergence curve for algorithms PSO, AccPSO (t = 0.1), APSOc, APSO for Beale function

Algorithm	Time (t)	Optimal Position	Optimal Fitness	Rank
PSO	-	['1.002913', '2.997653']	0.000015	1
AccPSO	0.05	['0.923576', '3.019822']	0.019049	4
AccPSO	0.1	['0.989945', '3.042390']	0.006080	3
AccPSO	0.25	['0.944149', '3.038851']	0.005785	2
AccPSO	0.5	['1.118404', '2.702696']	0.230431	7
AccPSO	1	['1.210614', '2.750605']	0.112572	5
AccPSO	5	['1.428052', '1.561826']	6.332959	9

APSOc	-	['0.897408', '3.234097']	0.134502	6
APSO	-	['0.559913', '3.351501']	0.348619	8

Table 6: Optimal Position, Optimal Fitness and Rank for Booth Function

From Table. 6 Rank column it can be observed that PSO performed best followed by AccPSO (t = 0.25) in second position. APSOc obtained sixth position. APSO obtained 8th Rank.

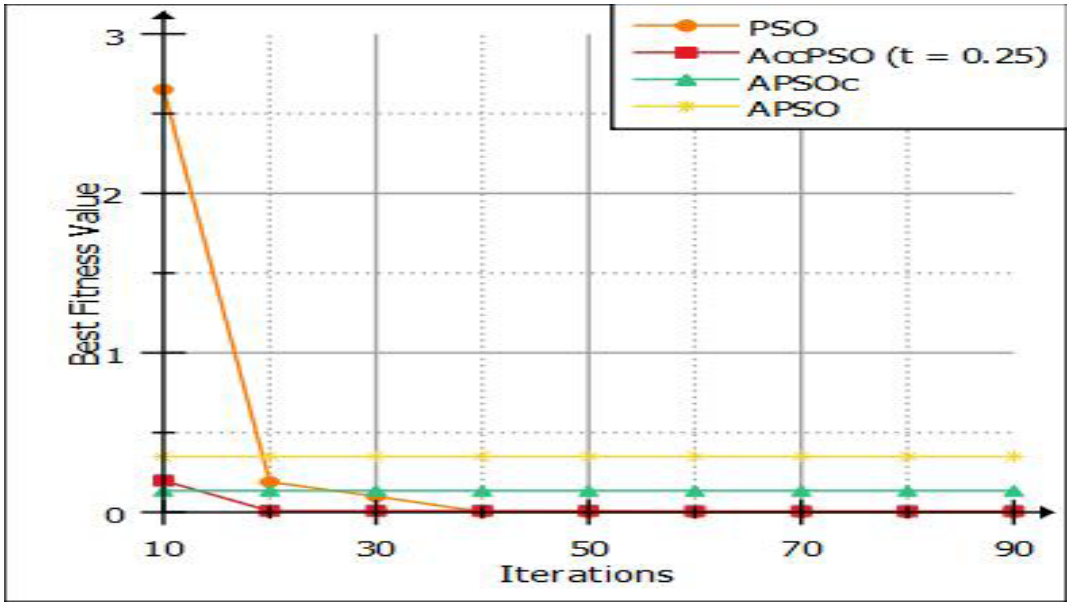

Figure 10: Convergence curve for algorithms PSO, AccPSO (t = 0.25), APSOc, APSO for Booth function

Algorithm	Time (t)	Optimal Position	Optimal Fitness	Rank
PSO	-	['0.029987', '-0.119793', '-2.790281']	0.012556	7
AccPSO	0.05	['-0.023015', '0.015646', '0.184435']	0.000944	4
AccPSO	0.1	['0.013953', '-0.007711', '-0.909649']	0.000341	2
AccPSO	0.25	['-0.017772', '0.026554', '3.144012']	0.000865	3
AccPSO	0.5	['0.014219', '-0.065331', '-2.529228']	0.003744	5
AccPSO	1	['-0.045747', '-0.001469', '-5.000000']	0.004250	6
AccPSO	5	['-1.761672', '1.509345', '0.358820']	0.694853	9
APSOc	-	['0.000000', '0.000000', '-5.000000']	0.000000	1
APSO	-	['-0.148320', '0.368496', '-7.057323']	0.124625	8

Table 7: Optimal Position, Optimal Fitness and Rank for Three-Hump Camel Function

From Table. 7 Rank column it can be observed that APSOc performed best followed by AccPSO (t = 0.1) in second position. PSO obtained Rank 7. APSO obtained Rank 8.

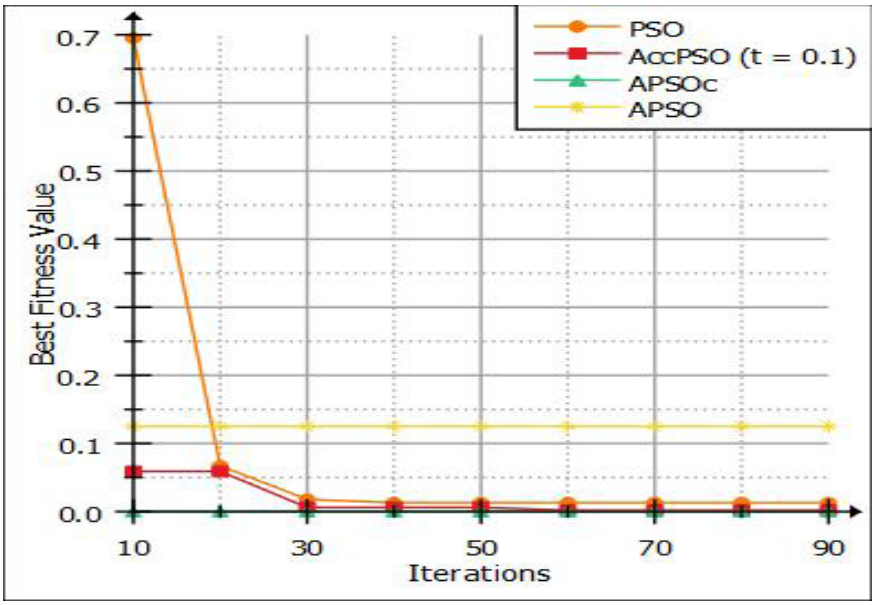

Figure 11: Convergence curve for algorithms PSO, AccPSO (t = 0.1), APSOc, APSO for Three-Hump Camel function

Function/ Rank	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Rank 6	Rank 7	Rank 8	Rank 9
Rastrigin	APSOc	PSO	AccPSO (t = 0.1)	AccPSO (t = 0.05)	AccPSO (t = 0.25)	AccPSO (t = 0.5)	AccPSO (t = 1)	AccPSO (t = 5) and APSO	None
Sphere	APSOc	AccPSO (t = 0.1)	PSO	AccPSO (t = 0.25)	AccPSO (t = 0.05)	APSO	AccPSO (t = 1)	AccPSO (t = 0.5)	AccPSO (t = 5)
Ackley	APSOc	AccPSO (t = 0.1)	AccPSO (t = 0.05)	AccPSO (t = 0.25)	PSO	AccPSO (t = 1)	AccPSO (t = 0.5)	APSO	AccPSO (t = 5)
Rosenbrock	PSO	AccPSO (t = 0.25)	AccPSO (t = 0.05)	AccPSO (t = 0.1)	APSOc	AccPSO (t = 1)	AccPSO (t = 0.5)	APSO	AccPSO (t = 5)
Beale	AccPSO (t = 0.1)	AccPSO (t = 0.05)	AccPSO (t = 0.5)	AccPSO (t = 0.25)	AccPSO (t = 1)	APSOc	AccPSO (t = 5)	PSO	APSO
Booth	PSO	AccPSO (t = 0.25)	AccPSO (t = 0.1)	AccPSO (t = 0.05)	AccPSO (t = 1)	APSOc	AccPSO (t = 0.5)	APSO	AccPSO (t = 5)
Three-Hump Camel	APSOc	AccPSO (t = 0.1)	AccPSO (t = 0.25)	AccPSO (t = 0.05)	AccPSO (t = 0.5)	AccPSO (t = 1)	PSO	APSO	AccPSO (t = 5)

Table 8: Ranking of Algorithms on Benchmark functions

From Table. 8 it can be concluded that APSOc performed best four times, PSO performed best 2 times and proposed AccPSO performed best one time for particular value of time "t".

In PSO literature, time and iterations are used interchangeably and generally time value "t" is 1. In this work novel AccPSO algorithm is proposed where time "t" is continuous value and can be varied. If we followed general rule of time "t" equal to 1 then we would have obtained only one single optimal value for AccPSO algorithm. In this work, it has been found that for AccPSO algorithm time "t" equals 0.1 obtained better results 5 times and time "t" equals 0.25 obtained better results 2 times when compared to time "t" equals 1 and other time values.

APSO proposed in [2] got Rank 8, Rank 6, Rank 8, Rank 8, Rank 9, Rank 9, Rank 8 respectively for 7 benchmark functions. A slight modification is introduced into APSO algorithm to create APSOc algorithm. APSOc is nothing but APSO with velocity and position clamping. If velocity crossed vmax then setting velocity to vmax and if velocity is less than vmin then setting velocity to vmin, this is Velocity clamping. Similarly, setting position between xmin and xmax when it crossed boundary is position clamping. APSOc (Slight modification of APSO proposed in [2]) got Rank 1, Rank 1, Rank 1, Rank 5, Rank 6, Rank 6, Rank 1 respectively for 7 benchmark functions. Hence it can be concluded that introducing velocity and position clamping into optimization algorithms can make a significant difference in the results obtained as observed in this work.

5. Conclusions

Results obtained show that APSOc performed best four times, PSO performed best 2 times and proposed AccPSO performed best one time for particular value of time "t". A novel Acceleration Particle Swarm Optimization (AccPSO) is proposed in this article. Unlike many PSO algorithms, AccPSO is based on calculating acceleration first. In AccPSO, time "t" is a continuous variable. If we change "t" value between 2 iterations we get different results. Hence a general rule of using iterations and time interchangeably and time "t" equals 1 between iterations is broken in AccPSO algorithm where time "t" is a continuous variable in this algorithm. It has been found that better results have been obtained for time "t" equals 0.1 five times and time "t" equals 0.25 two times when compared to time "t" equals 1 and other values. There is a significant improvement in APSO algorithm when clamping is introduced into it and APSO algorithm with clamping (APSOc) obtained Rank 1 four times. Hence it can be concluded that varying time "t" between iterations (unlike time "t" equals 1) may yield better results. Also introducing clamping into algorithm and checking results of algorithm with and without clamping may yield a significant difference in the results obtained just like results obtained by APSO and APSOc are different.

Declarations

All authors declare that they have no conflicts of interest.

References

1. Agrawal, J., & Agrawal, S. (2015). Acceleration based particle swarm optimization for graph coloring problem. Procedia Computer Science, 60, 714-721.
2. Shankar, A., Kandath, H., & Senthilnath, J. (2023, October). Acceleration-Based PSO for Multi-UAV Source-Seeking. In IECON 2023-49th Annual Conference of the IEEE Industrial Electronics Society (pp. 1-6). IEEE.
3. Agrawal, J., & Agrawal, S. (2015). Acceleration based particle swarm optimization (APSO) for RNA secondary structure prediction. In Progress in Systems Engineering: Proceedings of the Twenty-Third International Conference on Systems Engineering (pp. 741-746). Springer International Publishing.
4. Beheshti, Z., & Shamsuddin, S. M. H. (2014). CAPSO: centripetal accelerated particle swarm optimization. Information Sciences, 258, 54-79.
5. Paschos, A. E., Kapinas, V. M., Ntouni, G. D., Hadjileontiadis, L. J., & Karagiannidis, G. K. (2017, November). Dynamic spectrum sensing through accelerated particle swarm optimization. In 2017 25th Telecommunication Forum (TELFOR) (pp. 1-4). IEEE.
6. Beheshti, Z., Shamsuddin, S. M., Hasan, S., & Wong, N. E. (2016). Improved centripetal accelerated particle swarm optimization. Int. J. Advance Soft Compu. Appl, 8(2).
7. Zhang, H., & Yang, Z. (2018). Accelerated Particle Swarm Optimization to Solve Large-Scale Network Plan Optimization of Resource-Leveling with a Fixed Duration. Mathematical problems in engineering, 2018(1), 9235346.
8. KUMAR, Ashok et al. (2024). Parameter tuning for enhancing performance of a variant of particle swarm optimization algorithm. Indonesian Journal of Electrical Engineering and Computer Science, 36(2), 1253-1260.
9. Yao, J., Luo, X., Li, F., Li, J., Dou, J., & Luo, H. (2024). Research on hybrid strategy Particle Swarm Optimization algorithm and its applications. Scientific Reports, 14(1), 24928.
10. Tang, K., & Meng, C. (2024). Particle swarm optimization algorithm using velocity pausing and adaptive strategy. Symmetry, 16(6), 661.
11. Ahmed, G., Eltayeb, A., Alyazidi, N. M., Imran, I. H.,

Sheltami, T., & El-Ferik, S. (2024). Improved particle swarm optimization for fractional order PID control design in robotic Manipulator system: A performance analysis. Results in Engineering, 24, 103089.

12. Qiao, J., Wang, G., Yang, Z., Luo, X., Chen, J., Li, K., & Liu, P. (2024). A hybrid particle swarm optimization algorithm for solving engineering problem. Scientific Reports, 14(1), 8357.

13. Alao, M. A., Popoola, O. M., & Ayodele, T. R. (2024). An improved particle swarm optimisation algorithm for optimum placement and sizing of biogas-fuelled distributed generators for seasonal loads in a radial distribution network. Energy Reports, 12, 1531-1550.

14. Twumasi, E., Frimpong, E. A., Prah, N. K., & Gyasi, D. B. (2024). A novel improvement of particle swarm optimization using an improved velocity update function based on local best murmuration particle. Journal of Electrical Systems and Information Technology, 11(1), 42.

15. Dong, A., & Lee, S. K. (2024). The Study of an Improved Particle Swarm Optimization Algorithm Applied to Economic Dispatch in Microgrids. Electronics, 13(20), 4086.

16. Tarekegn Nigatu, D., Gemechu Dinka, T., & Luleseged Tilahun, S. (2024). Convergence analysis of particle swarm optimization algorithms for different constriction factors. Frontiers in Applied Mathematics and Statistics, 10, 1304268.

17. Siswanto, M., Ali, M., Haikal, M. A., Wahyudi, S., Soedarsono, S., & Djalal, M. R. (2024). Stability of Water Flow in Tanks Using Particle Swarm Optimization (PSO) Method. In E3S Web of Conferences (Vol. 473, p. 04003). EDP Sciences.

18. Liu, Y., Zhu, X., Zhang, X. Y., Xiao, J., & Yu, X. (2024). Rgg-pso+: Random geometric graphs based particle swarm optimization method for UAV path planning. International Journal of Computational Intelligence Systems, 17(1), 127.

19. Kazerani, R. (2024). Improving breast cancer diagnosis accuracy by particle swarm optimization feature selection. International Journal of Computational Intelligence Systems, 17(1), 44.

20. Soliman, H. Y., Megahed, A. A., Abdelazim, M., & Abdelhay, E. H. (2023). 5G sub-6 GHz wideband antenna with PSO optimized dimensions. Prog Electromagnet Res M, 120, 123-134.

21. Grandis, H., & Maulana, Y. (2017, April). Particle swarm optimization (PSO) for magnetotelluric (MT) 1D inversion modeling. In IOP conference series: Earth and Environmental Science (Vol. 62, No. 1, p. 012033). IOP Publishing.

22. Yu, P., Yi, J., Huang, T., Xu, Z., & Xu, X. (2024). Optimization of Transformer heart disease prediction model based on particle swarm optimization algorithm. arXiv preprint arXiv:2412.02801.

23. Syed, D., Shaikh, G. M., & Rizvi, S. (2024). Systematic review: particle swarm optimization (PSO) based load balancing for Cloud Computing. Sir Syed University Research Journal of Engineering & Technology, 14(1), 86-94.

24. Gajawada, S. (2025). Acceleration Particle Swarm Optimization. Available at SSRN 5138753.

25. Salem, H. S., Mead, M. A., & El-Taweel, G. S. (2024). Particle swarm optimization-based hyperparameters tuning of machine learning models for big COVID-19 data analysis. Journal of Computer and Communications, 12(3), 160-183.

26. K. Thamizhmaran. (2024). Different Effective Performance in Particle Swarm Optimization (PSO) Algorithm. Journal of Advancement in Electronics Design, 7(1), 11–16.

27. Li, Y. (2024). Particle swarm optimization based neural network automatic controller for stability steering control of four-wheel drive electric vehicle. Frontiers in Mechanical Engineering, 10, 1378175.