Investigating Boundary Constraint Handling Mechanisms in Particle Swarm Optimization

Quinton Weenink
dept. Computer Science
University of Pretoria
Pretoria, South Africa
u13176545@tuks.co.za

I. Introduction

This paper aims to investigate the effect of Boundary Constraint Handling Mechanisms on *particle swarm optimisation algorithm* (PSO). There are many approaches that claim to improve exploration of the viable search space. In comparing these approaches to the a PSO with no boundary constraints we could observe the changes such mechanisms are having on the performance of the PSO.

II. Particle Swarm Optimization

A PSO is a machine learning algorithm where each particle in the population represents a possible solution to the optimisation problem [1]. Each particle moves around the search space attempting to find the optimal solution with the influence of its neighbouring particles. The position for each particle iteration is calculated as follows.

$$\vec{x}_i^{\,t} = \vec{x}_i^{\,t-1} + \vec{v}_i^{\,t} \tag{1}$$

Where:

- $\vec{x}$ is the particle position vector in n-dimensions
- $\vec{v}$ is the particle velocity vector, used to calculate the particles next position
- t is the time increment
- i is the particle number

For each iteration of the algorithm, a new location of the particle is calculated based on its previous location and velocity vector as seen in (1). The following describes a the velocity update algorithm.

$$\vec{v}_i^{\,t} = w.\vec{v}_i^{\,t-1} + c_1.\vec{r}_1^{\,t}.(\vec{x}_{pBest,i} - \vec{x}_i^{\,t-1}) + c_2.\vec{r}_2^{\,t}.(\vec{x}_{nBest,i} - \vec{x}_i^{\,t-1}) \tag{2}$$

Where:

- c_1 is the acceleration coefficient for the cognitive component
- c_2 is the acceleration coefficient for the social component
- r_1 and r_2 is a vector of random numbers in the range (0, 1)
- $\vec{x}_{pBest}$ is the personal best position of that particle
- $\vec{x}_{nBest}$ is the best position found in that particles neighbourhood

Particle's social component, the third term in (2), requires $nBest$, the best neighbouring particles position. The set of neighbouring particles is determined by the topology used in the PSO. Different topologies could influence the performance of the PSO.

Additionally, velocity max, V_{max}, can be used as the limiter when calculating the particle's velocity in (2), ensuring that the particle stays inside search space as well as reduce skipping over more optimal solutions. V_{max} could however prevent particles from exploring more optimal solutions.

A. Social Network Topologies

In the above mentioned PSO the neighbourhood could be described in a variety of ways. One such method is the $gBest$ topology, where each particle is a neighbour of every other particle in the network. This means that the particle's $nBest$ is always the global best particle for the swarm.

Another social topology called $lBest$ connects particles in a ring topology, the neighbourhood is specified by a fixed size of n_s. The $nBest$ in this case is determined by the best particle within n_s neighbours from the current particle [2].

Finally, another social network topology exists called the *Von Neumann* (VN) social network topology. This topology connects each particle to its neighbours in a lattice [2]. In order to determine the $nBest$, the best error of its north, south, east and western, neighbours is used (2).

B. Benchmark functions

No shifted, noisy or rotated version of the functions were used. In an effort to make things scaleable different types of benchmarks functions were used in order to determine the fitness of the functions.

III. Boundary Constraint Handling Mechanisms

The following describes the boundary constraint handling mechanisms used in this research.

1) ***No boundary constraints****:* Particles were not constrained in any way.

TABLE I
BENCHMARK FUNCTIONS

Function	*Multimodal*	*Inseparable*
Absolute value		
Ackley	Yes	Yes
Hyperellipsoid		
Quartic		
Salomon	Yes	Yes
Schaffer 6	Yes	Yes

*2) **Feasible position update**:* Update the personal best positions only if the new particle position is better than its current personal best position, and if the new particle position is feasible. That is, a new particle position can not become a personal best position if it violates boundary constraints.

*3) **Clamping approach**:* If a particle violates a boundary constraint in a specific dimension, then clamp the corresponding decision variable at the boundary value.

*4) **Per element reinitialization**:* For any decision variable of any particle that violates a boundary constraint, reinitialize that decision variable to a random position that satisfies the boundary constraints.

*5) **Per element reinitialization and velocity = 0**:* Adapt the per element reinitialization approach above to also set the velocity of the decision variable that violates a boundary constraint to zero. The corresponding decision variables new position will therefore not be influenced by the momentum term.

*6) **Initialize to personal best position**:* Initialize the boundary violating decision variable to the corresponding personal best position.

*7) **Initialize to personal best position and velocity = 0**:* Adapt the intialize to personal best position strategy above to also set the corresponding velocity to 0.

*8) **Initialize to global best position**:* Initialize to global best position: As for the above, but rather the global best for the boundary violating decision variable.

*9) **Initialize to global best position velocity = 0**:* Initialize to global best position and set velocity to zero: Adapt the intialize to global best position strategy to also set the corresponding velocity to 0.

*10) **Reverse velocity**:* The velocity of the bounadry violating decision variable is simply reversed while that decision variable violates the boundary constraint.

*11) **Arithmetic average**:* Set the bounadry violating decision variable to an arithmetic average of the corresponding personal best and global best position.

IV. EXPERIMENTS

A. PSO configuration

In order to achieve viable results, each experiment was run as a mean over 50 samples. r_1 and r_2 sampled from a uniform distribution $(0, 1)$ as specified in equation 2.

A memory based PSO was used with a star or $gBest$ social topologie was used for this study. 30 particles were used for all functions. 30 dimensions were used for all functions. $c_1 = c_2 = 1.4$ as well $w = 0.7$

No V_{max} was used

V. RESULTS

TABLE II
RESULTS AFTER 5000 ITERATIONS. MEANS ARE REPORTED OVER 50 SAMPLES WITH STANDARD DEVIATIONS IN PARENTHESIS

	f_1	f_2	f_3
BC_1	12.576554695 (16.285050)	4.89108089516 (1.948212190)	6.6095267e-05 (0.00038360)
BC_2	5.1857243227 (9.6293110)	4.58116294703 (1.565766830)	2.786555e-06 (7.7739e-06)
BC_3	57.531250757 (65.651655)	6.00606880913 (3.434289275)	54.526024907 (90.3297955)
BC_4	8.2467633532 (11.461532)	4.55418774247 (1.648445767)	1.01489e-05 (4.589539e-05)
BC_5	5.1389339174 (9.0355906)	4.89810256890 (1.720959805)	9.465576e-05 (0.00032185)
BC_6	9.4929921868 (13.040311)	4.97768223959 (2.130050755)	9.924092e-06 (3.56359e-05)
BC_7	13.266182721 (22.017559)	5.07560687421 (2.252312710)	2.096943e-05 (9.81292e-05)
BC_8	15.164538450 (23.109397)	6.02058451060 (3.086329288)	0.0004165906 (0.00160546)
BC_9	20.925653391 (26.146772)	6.42808810347 (3.256848326)	0.0009752216 (0.00390649)
BC_10	599.58515676 (173.65772)	17.9263162102 (1.816182460)	797.0981965 (379.309663)
BC_11	20.710428245 (37.284614)	5.73874408816 (3.127739837)	15.008693567 (28.25022732)
	f_4	f_4	f_6
BC_1	1.9239946e-21 (1.08892e-20)	-0.259725655 (0.14964295)	9.913329509 (0.656834)
BC_2	1.7858202e-26 (1.1080e-25)	-0.1334162 (0.4525834)	9.194013340 (0.916300)
BC_3	0.37580965 (1.8605501)	0.2911377709212 (1.837814591)	10.4460088 (0.7901153)
BC_4	1.699433e-25 (8.329e-25)	-0.136358848 (0.32299095)	7.8171530590 (0.97061129)
BC_5	6.203499e-20 (4.2647e-19)	-0.134392611 (0.4148771)	9.17659654534 (1.1028635337)
BC_6	8.175363e-21 (5.722e-20)	-0.210002336 (0.28267089)	8.74905186834 (0.9779485313)
BC_7	2.418665e-16 (1.324e-15)	-0.0636568206 (0.466326789)	9.9507224427 (0.798154431)
BC_8	7.26727e-16 (5.086e-15)	-0.2007856567 (0.287498829)	10.1430737966 (0.852829383)
BC_9	1.907742e-18 (1.335e-17)	-0.053840835 (0.552251796)	9.899939307 (0.88794107)
BC_10	26.79745200 (15.81222)	15.5318252566 (2.559650039)	12.16846579 (0.50818004)
BC_11	4.786350e-16 (3.2479e-15)	-0.2006067722 (0.288737979)	10.158431877 (0.69947078)

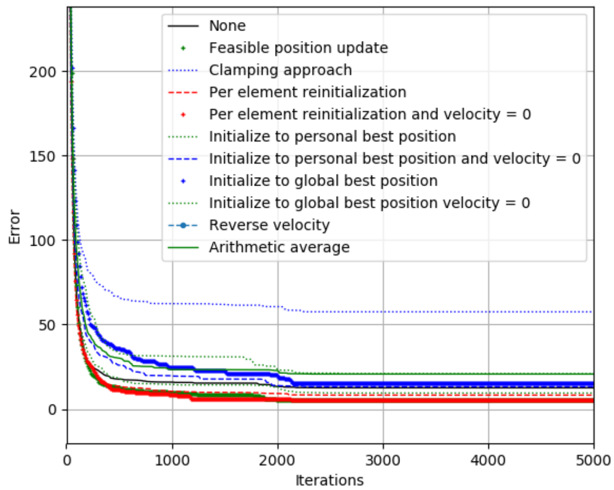

Fig. 1. f_1 performance

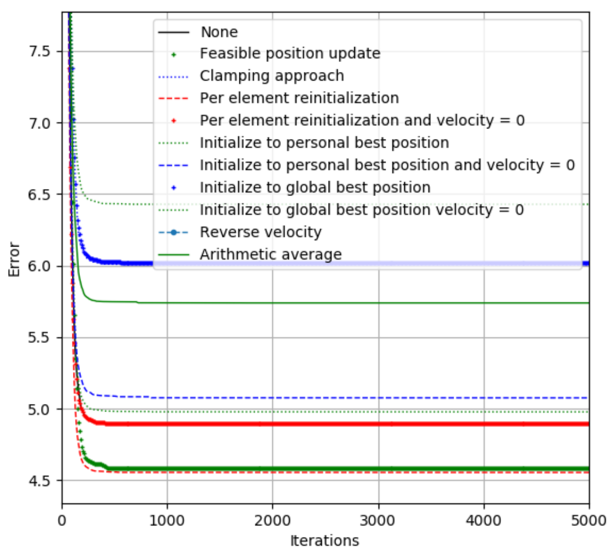

Fig. 2. f_2 performance

VI. Conclusion

In conclusion one can observe that that reversing the velocity performs quite poorly and almost immediately stops exploring. Clamping which one could have been assumed to be a viable approach seemed to perform poorly on all benchmark functions. Per-element initialisation to with out changing the velocity to zero introduces good results over all presumably due to the velocity allow it to still move towards its last found position.

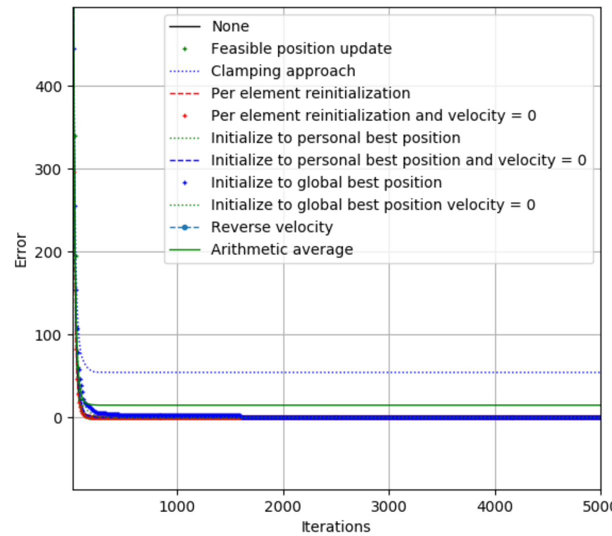

Fig. 3. f_3 performance

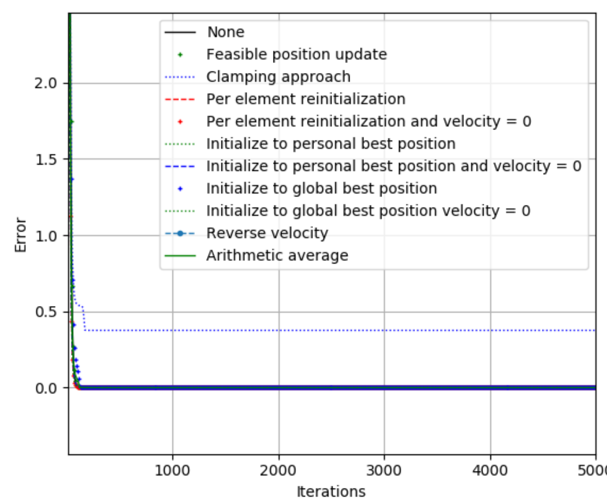

Fig. 4. f_4 performance

References

[1] A. Rakitianskaia and A. Engelbrecht, "Measuring saturation in neural networks," in *Symposium Series on Computational Intelligence*, IEEE. IEEE, Dec. 2015. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/7376778/

[2] A. B. van Wyk and A. P. Engelbrecht, "Overfitting by pso trained feedforward neural networks," in *IEEE Congress on Evolutionary Computation*, IEEE. IEEE, Jul. 2010. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/5586333/?section=abstract

[3] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

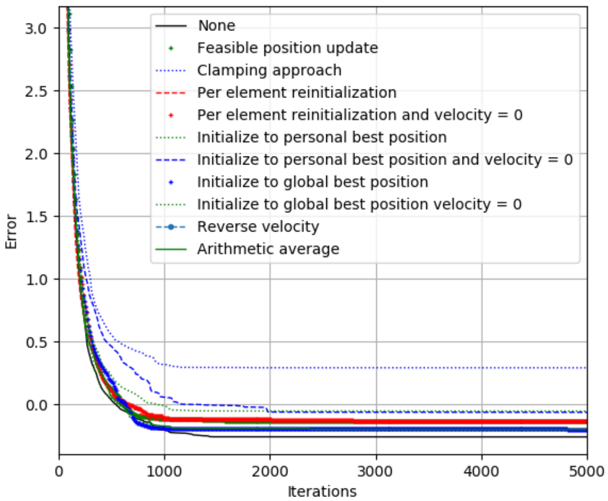

Fig. 5. f_5 performance

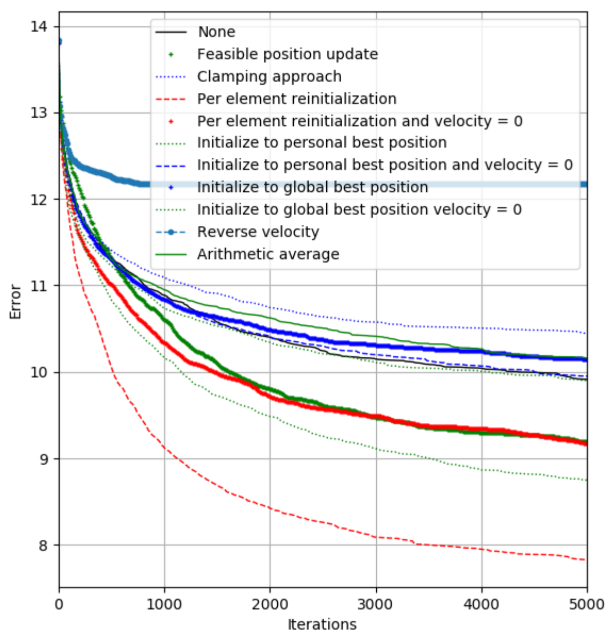

Fig. 6. f_6 performance

[4] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," *Micro Machine and Human Science*, vol. 5, no. 500, p. 3943, 1995. [Online]. Available: https://link.springer.com/article/10.1007/s00500-014-1222-z

VII. Appendix

f_1, the absolute value function, with $x_j \in [-100, 100]$

$$f(\mathbf{x}) = \sum_{j=1}^{n_x} |x_i| \tag{3}$$

f_2, the ackley function, with $x_j \in [-32.768, 32.768]$

$$f(\mathbf{x}) = -20e^{-0.2\sqrt{\frac{1}{n}\sum_{j=1}^{n_x} x_j^2}} - e^{\frac{1}{n}\sum_{j=1}^{n_x} cos(2\pi x_j)} + 20 + e \tag{4}$$

f_n, the hyperellipsoid function, with $x_j \in [5.12, 5.12]$

$$f(\mathbf{x}) = \sum_{j=1}^{n_x} j x_j^2 \tag{5}$$

f_n, the quartic function, with $x_j \in [5.12, 5.12]$

$$f(\mathbf{x}) = \sum_{j=1}^{n_x} j x_j^4 \tag{6}$$

f_n, the salomon function, with $x_j \in [-100, 100]$

$$f(\mathbf{x}) = -cos(2\pi \sum_{j=1}^{n_x} x_j^2) + 0.1\sqrt{\sum_{j=1}^{n_x} x_j^2} \tag{7}$$

f_n, the schaffer 6 function, with $x_j \in [-100, 100]$

$$f(x, y) = \sum_{j=1}^{n_x} \left(0.5 + \frac{sin^2(\sqrt{x^2 + y^2}) - 0.5}{[1 + 0.001 \cdot (x^2 + y^2)]^2} \right) \tag{8}$$