Investigating Chaotic Particle Swarm Optimisation for Neural Network Training

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

Supervisor:
Ms. Anna Bosman
dept. Computer Science
University of Pretoria
Pretoria, South Africa
annar@cs.up.ac.za

***Abstract*—Historically, particle swarm optimisation algorithms have been successfully applied to neural network training, sometimes outperforming traditional gradient-based approaches. Studies have however shown that particle swarm optimisation algorithms do not scale very well, performing poorly on high-dimensional neural network architectures. This study aims to investigate the effect of using the chaotic particle swarm optimisation algorithm for neural network training. Observing that chaotic maps do not always beneficially influence neural network training in a constantly significant manner.**

***Index Terms*—CPSO, NN, Chaotic**

I. Introduction

Chaotic maps are mathematical maps or evolution functions that exhibit some sort of chaotic behaviour. A *chaotic particle swarm optimisation algorithm* (CPSO) is a *particle swarm optimisation algorithm* (PSO) that makes use of a chaotic map as part of its implementation. There are different methods that all borrow from the same idea of applying the chaotic maps to the standard PSO algorithm [1], [2]. One of the best performing CPSO implementations uses a *chaotic pseudo-random number generator* (CPRNG) for the random numbers required by the PSO. The velocity for a particle is influenced by the chosen *random number generator* (RNG). Due to this, it can be hypothesised that the chosen RNG could have a significant influence on the PSO's performance [2].

While studies have been conducted on PSO trained *neural network* (NN)s [3]–[5], as well as the performance of CPSOs [2], [6], none have been found to investigate the influence of CPSO trained NNs. Performance as well as overfitting remains a gap in the field of CPSOs. While results do look positive for most applications of CPSOs these implementations were applied to specific situations that may not be a clear indication of CPSO being beneficial to all NN training.

This research aims to determine hwo the performance of a CPSO compaires to that of the PSO, described by Kennedy and Eberhart in [7], for NN training. In completing this research the understanding of CPSOs as well as their effectiveness they have on NN training will be improved. Additionally, this research aims to assist others when training NNs with CPSOs.

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 [5]. 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 [3].

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 [3]. In order to determine the $nBest$, the best error of its north, south, east and western, neighbours is used (2).

III. Chaotic Maps and CPRNGs

For this research, it was decided to implement the CPSO using a CPRNG due to previous studies indicating the better performance of such an implementation [2] [6]. Other implementations include using chaotic maps to place particles in the search space.

All the chaotic map's initial X_0 and Y_0 were sampled in a uniform distribution $(0, 1)$ unless stated otherwise.

Each of the following maps were iteratively sampled from the chaotic maps listed bellow over 5000 iterations. Each batch was then scaled from (0, 1) as required by the PSO for r_1 and r_2 in Equation (2).

A. Tinkerbell map

The Tinkerbell map (depicted in Fig. 1) is a two-dimensional complex discrete-time dynamical system given by Eqs. (3) and (4).

Due to the potentially exponential nature of the algorithm the initial X_n and Y_0 had to be randomly distributed to $-(0.01, 0.1)$ and $(0, 0.1)$ respectively.

$$X_{x+1} = X_n^2 - Y_n^2 + aX_n + bY_n \tag{3}$$

$$Y_{n+1} = 2X_nY_ncX_n + dY_n \tag{4}$$

The following parameters were used in this research: $a = 0.9$ and $b = -0.6$ used in (3) as well as $c = 2$ and $d = 0.5$ used in (4) as suggested in [2].

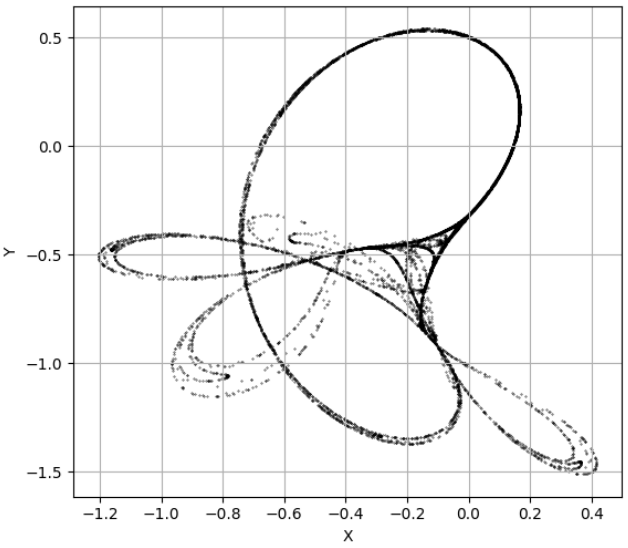

Fig. 1. Tinkerbell map

B. Lozi map

Depicted in Fig. 2, the Lozi map is a simple discrete two-dimensional chaotic map. The map equations are given in 5 and 6.

$$X_{x+1} = 1 - a|X_n| + bY_n \tag{5}$$

$$Y_{n+1} = X_n \tag{6}$$

The parameters used in (5) for this work are: $a = 1.7$ and $b = 0.5$ as suggested in [2].

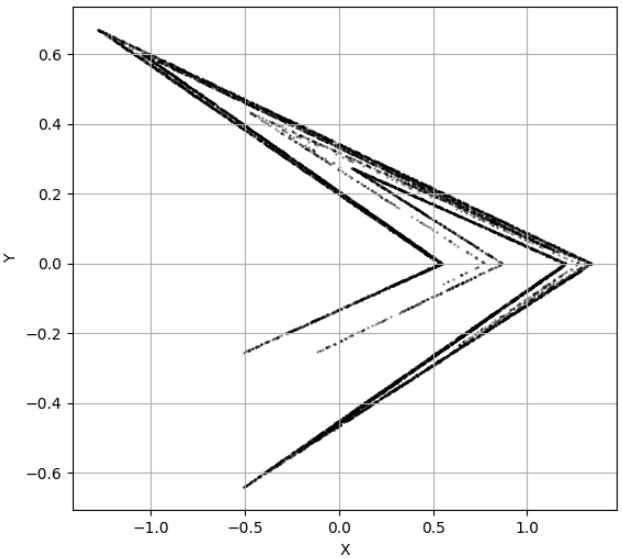

Fig. 2. Lozi map

C. Dissipative standard map

The Dissipative standard map is a two-dimensional chaotic map [2]. The Dissipative standard map is given in Fig. 3 with the map equations are given in Eqs. (7) and (8).

Observing the results of [1] one can hypothesise that the Dissipative standard map would perform better in higher dimentional problems.

$$X_{x+1} = X_n + Y_{n+1}(mod 2\pi) \tag{7}$$

$$Y_{n+1} = bY_n + ksinX_n(mod 2\pi) \tag{8}$$

The constants used in this work are $b = 0.6$ and $k = 8.8$ in (8) based on previous experiments [2].

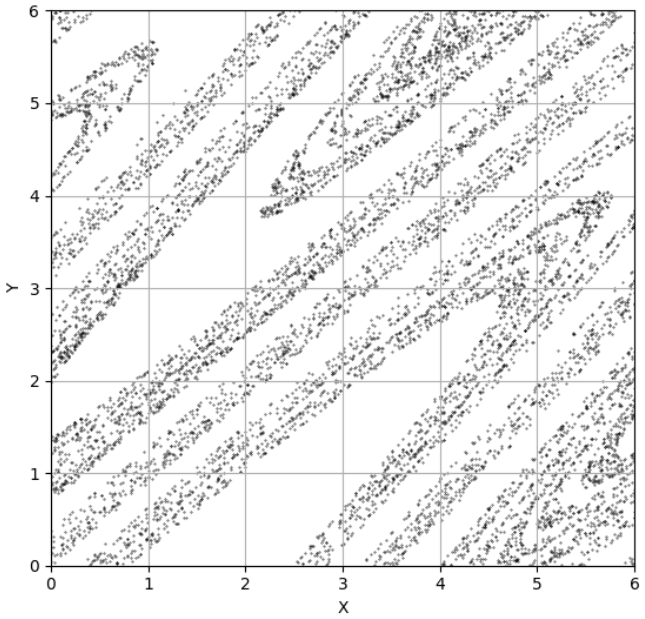

Fig. 3. Dissipative standard map

IV. CPSO

Historically, PSOs have been successfully applied to NN training, sometimes outperforming traditional gradient-based approaches [4], [5]. Studies have however shown that PSOs do not scale very well, performing poorly on high-dimensional NN architectures. This is as a result of high-dimensionality of the PSO's search space, an inherent property of NNs where the number of weights increases exponentially with the linear increase of architecture [4].

A CPSO is a PSO that makes use of the CPRNG to generate $\vec{r}_1$ and $\vec{r}_2$ when calculating the particle's velocity in (2). The CPRNG will use a chaotic map in order to calculate the generate the random number vectors.

While the work by Pluhacek et al. has proven that CPSOs offer better performance over standard PSOs in a number of cases [2], [6], [8], it is not safe to conclude that PSO NN training will be improved in general. Additionally, research has not explored precisely when to use which chaotic map. Some research [6], [8] attempts to solve this problem by using a multi-CPSO or an ensemble of CPSOs with the aim to design universally usable CPSOs that will not suffer from some disadvantages of previous CPSO designs [9].

A. Neural Network

Typically, feedforward NNs are comprised of neurons connected in layers. Signals travel from the input layer through the hidden layer(s) to the output layer [5]. Each neuron is connected through a collection of weights to the neurons in the next layer as can be seen in Fig. 4. This study used a NN with bias on every layer except the output layer.

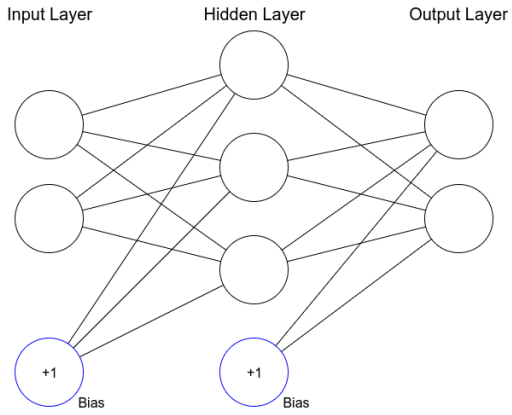

Fig. 4. Neural network with bias

Back propagation is an often used training method for NNs and makes use of forward propagation to generate the NN's output value(s). The weights are manipulated to train the network using the training error.

$$g(net) = \frac{1}{1 - e^{-net}} \tag{9}$$

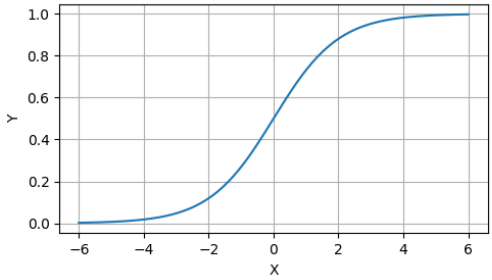

Fig. 5. Sigmoid activation function

Fig. (5) represents the output of a sigmoid function (9) which will be used as the activation function for NNs in this this research.

B. PSO trained NN

The goal of training a NN with a PSO is to find an optimal set of weights for the NN. The NNs weights are represented by the PSO's dimensions. Each particle represents a NN

which could be a possible solution. The particle's fitness is determined by the accuracy of its resulting NN.

TABLE I
PSO DIMENTIONS FOR EACH DATA SET

Data Set	*Dimensions*
Glass Identification	198
Iris	67
Wine	173
Prima Indians Diabetes	222
Cleveland Heart	195

This research hopes to determine weather the influence of a CPRNG allows the trained NN to perform better even when NN dimensions increase. The number of dimensions each of the particles will navigate is depicted in Table I.

C. Data sets selection and preparation

The selection of data sets were based on previous studies that were done on PSO training NN performance [3], [4]. Well known and documented previous studies that show the resulting performance on that data set will also been chosen as a secondary control to measure performance against.

All input to the NN was scaled to the range $[-1, 1]$. While all classification output was mapped to either 0.1 or 0.9.

D_T was measured as $\frac{2}{3}$ of the total data set and the remainder was allocated to D_G.

TABLE II
NEUAL NETWORK CONFIGURATION FOR DATA SETS

Data Set	*Input*	*Hidden*	*Output*	*Source*
Glass Identification	9	12	6	UCI MLR [10]
Iris	4	8	3	UCI MLR [10]
Wine	13	10	3	UCI MLR [10]
Pima Indians Diabetes	8	20	2	UCI MLR [10]
Cleveland Heart	13	10	5	UCI MLR [10]

V. MEASURING PERFORMANCE

NN accuracy is used to determine whether CPSO training has improved results over standard PSO training. The training error, E_T, was used in order to assess the resulting NN accuracy. The training error is calculated as the *mean squared error* (MSE) over the training set D_T of size P_T represented in (10) [3].

$$E_T = \frac{\sum_{p=1}^{P_T} \sum_{k=1}^{K} (t_{k,p} - o_{k,p})^2}{P_T K} \tag{10}$$

Similarly to the training error the generalisation error, E_G, can be used to calculate the performance of the NN on the set D_G.

Additionally the generalisation factor $\rho_F = \frac{E_G}{E_T}$ was also used to measure the overfitting of the trained NNs. Each NN was trained over 5000 iterations of the PSO's algorithm. A $\rho_F > 1$ would indicate a possibility of overfitting due to the generalisation error being greater than the training error. While a $\rho_f < 1$ or $E_G < E_T$ could indicate a lack of overfitting.

VI. EXPERIMENTS

A. CPRNG configuration

In order to achieve viable results, each experiment was run as a mean over 30 samples. A control was run with a PSO that uses a standard RNG with r_1 and r_2 sampled from a uniform distribution $(0, 1)$ as specified in equation 2. The overfitting and performance of the control was measured as a reference. The CPSOs will also be observed in the same manner.

For each result observed the only change was a different RNG and therefor any significant repeatable performance change measured could be assumed to be the result of this change.

B. PSO configuration

Both the VN and $gBest$ social network topologies were used for this study. The $gBest$ PSO used 25 particles, while the VN PSO used a 5 by 5 lattice of 25 particles.

VII. RESULTS

The resulting E_T, E_G and ρ_F for all experiments conducted in this study are given in Tables III, IV, V, VI and VII.

Observing ρ_F over the results one can see the CPSOs, while often outperforming PSO, result in a NN with higher overfitting. A good depiction of this is the CPSO's performance on the Iris problem. The randomly sampled PSO does not perform as well in training, but, results in a NN that appears to have less overfitting.

CPSOs can be observed to perform well for problems that have a higher error at the end of training, this is were some of the CPSO's performance become more evident. As can be observed in result Tables, the smaller the error the less likely the CPSOs will result in a distinguishable performance increase over the randomly sampled PSO.

Over all results the Lozi and Tinkerbell chaotic maps CPSO perform best on the problems resulting in a NNs with less overfitting and higher E_G than the Dissipative standard map. The dissipative standard map did not perform well in most circumstances contrary what was hypothesised, performing poorly in overall.

The results in Table III, specifically the results where $V_{max} = 0.1$ is applied with a VN topology, represent a relatively consistent performance increase across most problems for CPSOs over uniform PSOs. The Lozi map CPSO outperforms the basic PSO as well as the Tinkerbell and Dissipative map CPSO.

Both Tables III and IV are a good case for observing the converse of above. The basic PSO almost always outperforms all CPSOs across almost all data sets. This again pertains to the experiment using the VN topology but differs with the use of no V_{max} restriction.

Table VII one of the best cases for the CPSO with a better performance of both Tinkerbell and Lozi.

Figures 6, 7, 8, 9, and 10 were chosen based on weather a distinguishable observation could be made.

Fig. 8 represents another more distinguishable result of this study where a CPSO's performance befit can be properly

TABLE III
GLASS RESULTS AFTER 5000 ITERATIONS. MEANS ARE REPORTED OVER 30 SAMPLES WITH STANDARD DEVIATIONS IN PARENTHESIS

$gBest$ topology					
		Tinkerbell	***Lozi***	***Dissipative***	***Uniform***
No V_{max}	E_T	0.04525184 (0.008998)	0.04510496 (0.015825)	**0.04380078** (0.011896)	0.04924612 (0.014376)
	E_G	**0.07788302** (0.006962)	0.07928021 (0.011482)	0.07839134 (0.012285)	0.08072251 (0.017275)
	ρ_F	1.76569666 (0.267126)	1.88129248 (0.444181)	1.87329206 (0.433689)	**1.69060593** (0.307194)
$V_{max} = 0.1$	E_T	0.02331139 (0.003019)	**0.02245534** (0.002702)	0.02349460 (0.002779)	0.02298055 (0.002288)
	E_G	0.05786972 (0.002911)	0.05928630 (0.007112)	**0.05754601** (0.005912)	0.05945405 (0.005653)
	ρ_F	2.52469112 (0.350592)	2.68452969 (0.486994)	**2.49401208** (0.443459)	2.62478755 (0.455448)
Von Neumann topology					
		Tinkerbell	***Lozi***	***Dissipative***	***Uniform***
No V_{max}	E_T	0.06762034 (0.009678)	0.06843397 (0.006862)	0.07169069 (0.012895)	**0.06717697** (0.008959)
	E_G	0.08771098 (0.013657)	0.08637538 (0.009285)	0.09128531 (0.014901)	**0.08421155** (0.013971)
	ρ_F	1.31367391 (0.215528)	1.27970440 (0.222547)	1.29537791 (0.223924)	**1.26422974** (0.207465)
$V_{max} = 0.1$	E_T	0.06103854 (0.004123)	**0.06080808** (0.005100)	0.06242042 (0.004804)	0.06300097 (0.006018)
	E_G	0.08317552 (0.007380)	**0.08229008** (0.008072)	0.08453649 (0.007668)	0.08566786 (0.010444)
	ρ_F	1.37143484 (0.169578)	**1.36674597** (0.203229)	1.36708748 (0.196021)	1.37221453 (0.214361)

TABLE IV
IRIS RESULTS AFTER 5000 ITERATIONS. MEANS ARE REPORTED OVER 30 SAMPLES WITH STANDARD DEVIATIONS IN PARENTHESIS

$gBest$ topology					
		Tinkerbell	***Lozi***	***Dissipative***	***Uniform***
No V_{max}	E_T	**0.00431351** (0.001897)	0.00482629 (0.002183)	0.00535656 (0.002888)	0.01033975 (0.021587)
	E_G	0.02731797 (0.013110)	**0.02433567** (0.008977)	0.02660090 (0.013168)	0.02743795 (0.027432)
	ρ_F	7.26520511 (3.846323)	7.16758581 (5.888420)	6.56600247 (5.101546)	**6.56074316** (5.836499)
$V_{max} = 0.1$	E_T	0.00214609 (0.001303)	**0.00174290** (0.001365)	0.00177659 (0.001019)	0.00253293 (0.001785)
	E_G	0.02028782 (0.008862)	0.01923971 (0.007656)	0.01989378 (0.008261)	**0.01713263** (0.006033)
	ρ_F	14.9864899 (11.49917)	20.0014144 (16.22635)	16.5758342 (13.25325)	**13.4835849** (13.07027)
Von Neumann topology					
		Tinkerbell	***Lozi***	***Dissipative***	***Uniform***
No V_{max}	E_T	0.02306107 (0.009922)	0.02041756 (0.008811)	0.02065217 (0.005960)	**0.01862638** (0.006903)
	E_G	0.02506312 (0.013233)	0.02369337 (0.009089)	0.02377297 (0.006519)	**0.01938869** (0.007605)
	ρ_F	1.11893239 (0.493708)	1.25016177 (0.443750)	1.21050577 (0.361705)	**1.11046472** (0.414793)
$V_{max} = 0.1$	E_T	0.00921532 (0.003167)	0.00833094 (0.003190)	**0.00829776** (0.003227)	0.00886477 (0.003080)
	E_G	**0.01661334** (0.007306)	0.01842011 (0.007896)	0.01765593 (0.008138)	0.01706100 (0.007613)
	ρ_F	**2.61792201** (2.726593)	3.09107570 (2.736109)	3.07521083 (2.775890)	2.68828159 (2.436320)

TABLE V
WINE RESULTS AFTER 5000 ITERATIONS. MEANS ARE REPORTED OVER 30 SAMPLES WITH STANDARD DEVIATIONS IN PARENTHESIS

$gBest$ topology					
		Tinkerbell	***Lozi***	***Dissipative***	***Uniform***
No V_{max}	E_T	**0.00447088** (0.002862)	0.00506288 (0.001890)	0.00562712 (0.003118)	0.00449492 (0.003527)
	E_G	0.04408335 (0.018709)	0.04304486 (0.015388)	0.04197183 (0.018403)	**0.04257008** (0.014548)
	ρ_F	27.0083903 (55.44730)	**10.2892335** (6.925383)	15.1338175 (27.61655)	66.8660072 (192.1733)
$V_{max} = 0.1$	E_T	2.62E-05 (2.19E-05	2.05E-05 (1.54E-05	**1.81E-05** (1.71E-05	2.33E-05 (2.74E-05
	E_G	0.01051495 (0.005716)	0.01072558 (0.006103)	**0.00930577** (0.005016)	0.00994892 (0.004687)
	ρ_F	4042.90987 (11724.04)	**1131.28303** (1292.450)	4393.26135 (9666.408)	3328.85793 (6179.607)
Von Neumann topology					
		Tinkerbell	***Lozi***	***Dissipative***	***Uniform***
No V_{max}	E_T	0.03138550 (0.035686)	0.02028005 (0.032514)	0.01592315 (0.035002)	**0.00854521** (0.008621)
	E_G	0.06016326 (0.040704)	0.04769967 (0.041827)	0.04397668 (0.035265)	**0.03911403** (0.026738)
	ρ_F	**7.50539998** (9.273277)	10.0865305 (12.55283)	11.6433532 (18.42755)	12.2053416 (21.99583)
$V_{max} = 0.1$	E_T	5.61E-06 (1.05E-05)	1.02E-05 (2.52E-05)	2.61E-06 (3.35E-06)	**2.29E-06** (3.66E-06)
	E_G	**0.01013939** (0.005753)	0.01019491 (0.005391)	0.01103547 (0.005529)	0.01054944 (0.004678)
	ρ_F	396623.958 (1355246)	**31129.3501** (49291.88)	107577.308 (200753.4)	50478.8035 (91026.82)

TABLE VI
DIABETES RESULTS AFTER 5000 ITERATIONS. MEANS ARE REPORTED OVER 30 SAMPLES WITH STANDARD DEVIATIONS IN PARENTHESIS

$gBest$ topology					
		Tinkerbell	***Lozi***	***Dissipative***	***Uniform***
No V_{max}	E_T	**0.06608267** (0.006430)	0.06966550 (0.005228)	0.07059256 (0.012258)	0.07020193 (0.007254)
	E_G	0.12510246 (0.010445)	**0.12217592** (0.016235)	0.12969946 (0.019731)	0.13055759 (0.019258)
	ρ_F	1.92096131 (0.313345)	**1.77013492** (0.310815)	1.87666195 (0.372226)	1.87566668 (0.307047)
$V_{max} = 0.1$	E_T	0.06277231 (0.005429)	**0.06119740** (0.004161)	0.06237739 (0.004909)	0.06295500 (0.002947)
	E_G	0.12101642 (0.013324)	0.11937025 (0.010940)	**0.11840145** (0.010898)	0.11847174 (0.008573)
	ρ_F	1.95772470 (0.369080)	1.96872699 (0.300580)	1.91973813 (0.312562)	**1.88981650** (0.201324)
Von Neumann topology					
		Tinkerbell	***Lozi***	***Dissipative***	***Uniform***
No V_{max}	E_T	**0.13712405** (0.011292)	0.13736940 (0.011304)	0.14156675 (0.008928)	0.13895461 (0.007242)
	E_G	**0.17290003** (0.017876)	0.17580463 (0.019387)	0.17879292 (0.020631)	0.17505954 (0.017384)
	ρ_F	1.27755209 (0.222987)	1.29596902 (0.228948)	**1.27365954** (0.206006)	1.26786805 (0.175837)
$V_{max} = 0.1$	E_T	**0.12915034** (0.004476)	0.13010144 (0.004300)	0.13274713 (0.003209)	0.13140528 (0.002986)
	E_G	0.16122364 (0.005959)	0.16238036 (0.006410)	**0.16028312** (0.005631)	0.16164381 (0.007306)
	ρ_F	1.25122088 (0.088366)	1.25059195 (0.083359)	**1.20893112** (0.068054)	1.23173710 (0.079888)

TABLE VII
HEART RESULTS AFTER 5000 ITERATIONS. MEANS ARE REPORTED OVER 30 SAMPLES WITH STANDARD DEVIATIONS IN PARENTHESIS

$gBest$ topology					
		Tinkerbell	***Lozi***	***Dissipative***	***Uniform***
No V_{max}	E_T	0.06527673 (0.010578)	**0.05985821** (0.012627)	0.07079662 (0.006479)	0.06877290 (0.009432)
	E_G	0.11504191 (0.011177)	**0.10999252** (0.019076)	0.11292785 (0.012274)	0.11537514 (0.009757)
	ρ_F	1.80094297 (0.276929)	1.91099053 (0.496232)	**1.61821149** (0.294817)	1.70668156 (0.248506)
$V_{max} = 0.1$	E_T	0.02893253 (0.002502)	0.02864481 (0.003695)	**0.02806637** (0.004105)	0.02817930 (0.003654)
	E_G	**0.09769138** (0.009257)	0.10041339 (0.010432)	0.10115701 (0.006607)	0.10023158 (0.011999)
	ρ_F	**3.40709262** (0.462944)	3.59017091 (0.726701)	3.69959889 (0.696566)	3.65089603 (0.795225)
Von Neumann topology					
		Tinkerbell	***Lozi***	***Dissipative***	***Uniform***
No V_{max}	E_T	0.10614555 (0.007277)	**0.10019775** (0.006670)	0.10110355 (0.006069)	0.10138598 (0.006259)
	E_G	0.13160459 (0.008005)	**0.12488599** (0.008044)	0.12720097 (0.009739)	0.12684580 (0.011518)
	ρ_F	**1.24656109** (0.121877)	1.25042597 (0.098970)	1.26487493 (0.143885)	1.25760492 (0.152415)
$V_{max} = 0.1$	E_T	0.06493084 (0.004042)	0.06436446 (0.003085)	**0.06427986** (0.005968)	0.06490448 (0.005144)
	E_G	**0.11887931** (0.009025)	0.12036128 (0.006750)	0.12165407 (0.009391)	0.11948626 (0.009669)
	ρ_F	**1.84041675** (0.202234)	1.87502154 (0.150929)	1.92101234 (0.324871)	1.86392564 (0.298255)

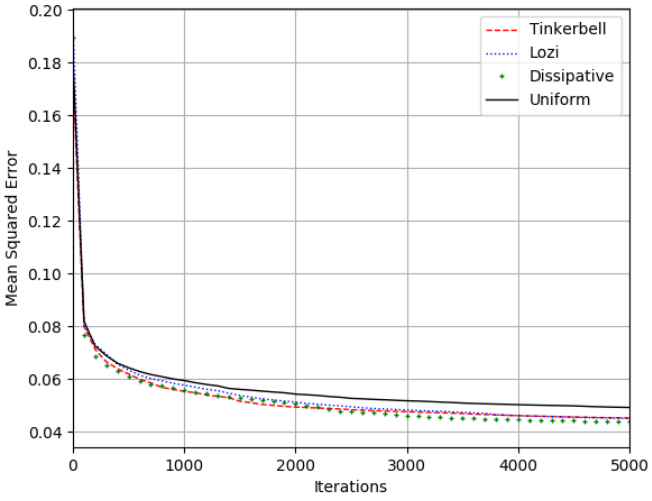

Fig. 6. $gBest$ topology- Glass Identification - E_T over 5000 iterations - No V_{max}

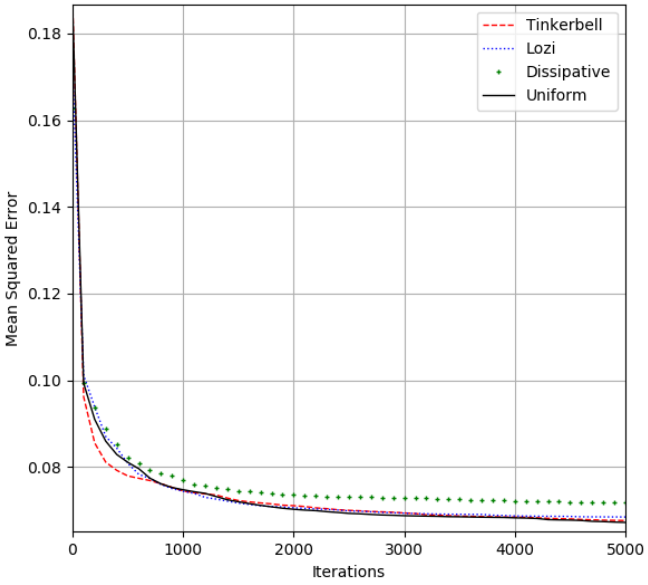

Fig. 7. VN topology- Glass Identification - E_T over 5000 iterations - No V_{max}

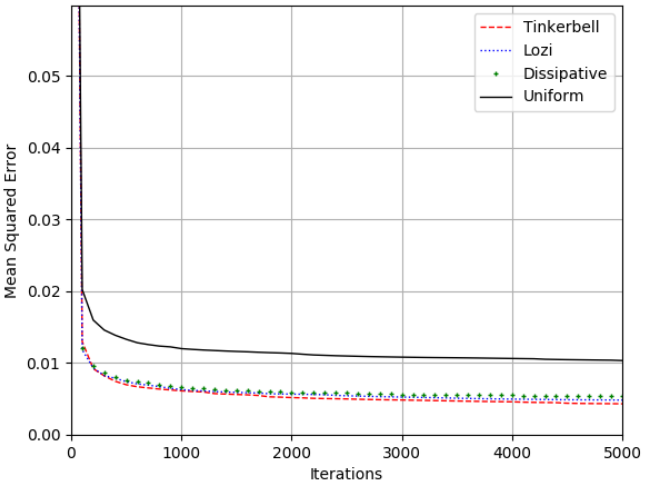

Fig. 8. $gBest$ topology- Iris - E_T over 5000 iterations - No V_{max}

observed. The uniform PSO is seen performing significantly worse than the CPSO sampled from Tinkerbell, Lozi and Dissipative. While this result appears positive this is not an the general tendency over all experiments as can be observed in Fig. 7 and 9.

The uniformly sampled PSO in Fig. 10 can be seen outperforming the other chaotic maps.

Lozi and Tinkerbell can be observed performing well in training in Fig. 10.

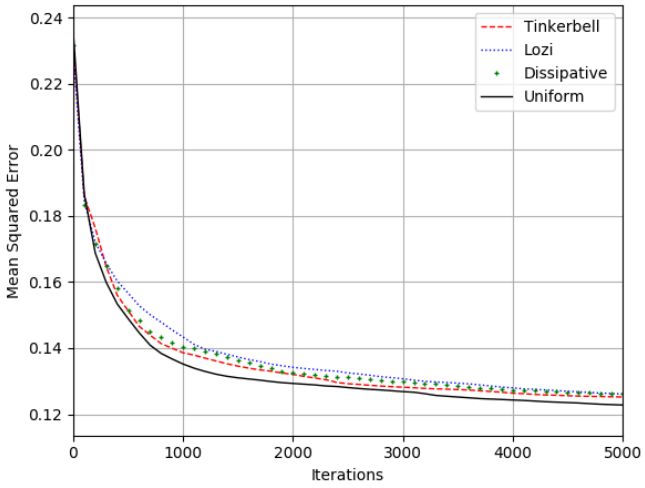

Fig. 9. VN topology- Diabetes - E_T over 5000 iterations - No V_{max}

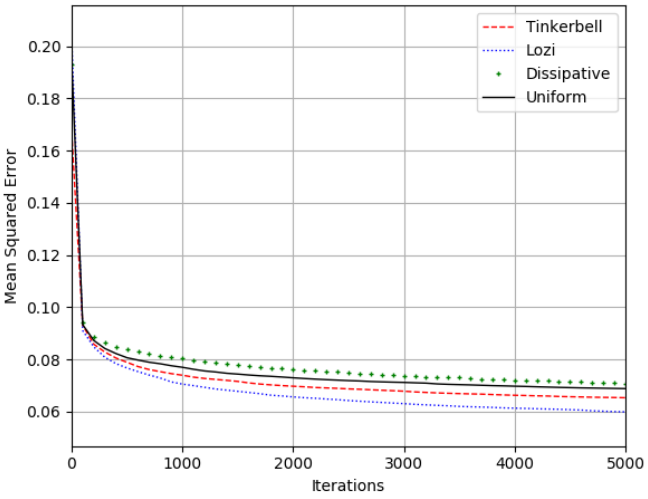

Fig. 10. $gBest$ topology- Heart - E_T over 5000 iterations - No V_{max}

VIII. Conclusion

In conclusion it is difficult to recommend a CPSO as an improvement over a basic PSO for NN training. While a CPSO does often allow for reduced NN training and generalisation error in some cases this results in increased overfitting. Additionally CPSOs may occasionally reduce training and generalisation error. A Basic PSO sampled uniformly form $(0, 1)$ resulted in performance difference that were statistically insignificant from a CPSO in a significant amount of experiments.

Further research could involve applying CPSOs to problems that require significantly higher number of weights.

References

[1] M. Pluhacek, R. Senkerik, D. Davendra, Z. K. Oplatkova, and I. Zelinka, "On the behavior and performance of chaos driven pso algorithm with inertia weight," *Computers & Mathematics with Applications*, vol. 66, no. 2, p. 122134, Aug. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0898122113000333

[2] M. Pluhacek, R. Senkerik, I. Zelinka, and D. Davendra, *Chaos Driven PSO On the Influence of Various CPRNG Implementations An Initial Study*, 1st ed., ser. 2194-7287. Springer International Publishing, 2014, vol. 14.

[3] 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

[4] A. Rakitianskaia and A. Engelbrecht, "Saturation in pso neural network training: Good or evil?" in *Congress on Evolutionary Computation (CEC)*, IEEE. IEEE, May 2015. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/7256883/

[5] 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/

[6] M. Pluhacek, R. Senkerik, and D. Davendra, "Chaos particle swarm optimization with eensemble of chaotic systems," *Swarm and Evolutionary Computation*, vol. 25, pp. 29–35, Dec. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2210650215000838

[7] 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

[8] M. Pluhacek, R. Senkerik, and I. Zelinka, "Multi-swarm aproach for chaotic pso algorithm - an initial study," in *MENDEL 2014*, ser. January, vol. 2014-January, 20th International Conference on Soft Computing: Evolutionary Computation, Genetic Programming, Swarm Intelligence, Fuzzy Logic, Neural Networks, Fractals, Bayesian Methods, MENDEL 2014. Brno: Brno University of Technology, 2014, pp. 141–146.

[9] M. Pluhacek, R. Senkerik, and I. Zelinka, "Particle swarm optimization algorithm driven by multichaotic number generator," *Soft Computing*, vol. 18, no. 500, p. 631639, 2014. [Online]. Available: https://link.springer.com/article/10.1007/s00500-014-1222-z

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

[11] M. Pluhacek, R. Senkerik, I. Zelinka, and D. Davendra, *Tuning the Lozi Map in Chaos Driven PSO Inspired by the Multi-chaotic Approach*, 1st ed., ser. Advances in Intelligent Systems and Computing. Springer International Publishing, 2014, vol. 289.

[12] Y. Song, Z. Chen, and Z. Yuan, "New chaotic pso-based neural network predictive control for nonlinear process," *IEEE Transactions on Neural Networks*, vol. 18, no. 2, pp. 595 – 601, Mar. 2007. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/4118283/