An Exploration of Gradually Increasing Directed Neighbourhoods for Particle Swarm Optimisation

Abstract

Particle swarm optimisation (PSO) is an intelligent random search algorithm, and the key to success is to effectively balance between the exploration of the solution space in the early stages and the exploitation of the solution space in the late stages. This paper presents a new dynamic topology called "gradually increasing directed neighbourhoods (GIDN)" that provides an effective way to balance between exploration and exploitation in the entire iteration process. In our model, each particle begins with a small number of connections and there are many small isolated swarms that improve the exploration ability. At each iteration, we gradually add a number of new connections between particles which improves the ability of exploitation gradually. Furthermore, these connections among particles are created randomly and have directions. We formalise this topology using random graph representations. Experiments are conducted on 31 benchmark test functions to validate our proposed topology. The results show that the PSO with GIDN performs much better than a number of the state of the art algorithms on almost all of the 31 functions.

Keywords:
Particle Swarm Optimisation, Dynamic Topologies, Neighbourhood Topologies, Exploration and Exploitation

1. Introduction

Particle Swarm Optimisation (PSO) was proposed by Eberhart and Kennedy in 1995 [1, 2]. It is inspired by the socially self-organised populations such as bird flocking and fish schooling. The PSO algorithms (PSOs) have gained increasing popularity in recent years. The PSOs have been widely used in many science and engineering domains [3, 4]. This is mainly due to its fast convergence rate and few parameters to tune.

The vital property of the PSO is the interactions or connections between particles. These connections are generally known as the "neighbourhood topology" of

the algorithm. The neighbourhood topologies of the swarm determine the speed of information flow in the entire population, and furthermore, the speed of information flow could be used to control exploration and exploitation of the search space [5]. The most commonly used topologies are $gBest$ (or the fully connected topology) and $lBest$ (or the ring topology) [6]. In the $gBest$ topology, all the particles are connected, and consequently at each iteration, the best position attracts all the particles towards that location. The information flow in the $gBest$ is extremely fast. The PSO with the $gBest$ topology has strong ability on exploitation (the use of known solutions), but could be easily trapped into local optima and cause the premature convergence problem. As for the $lBest$, each particle is only directly connected with two adjacent neighbours, and therefore the information flow spreads around the population quite slowly, which improves the PSO's ability on exploration (the search for new solutions). However, it eventually affects the convergence speed of the PSO and usually requires a large number of objective function evaluations.

Therefore, we could say that the $gBest$ and $lBest$ topologies represent the two extremes of the information flow speed. But we could also state that the $lBest$ topology meets the requirements of exploration in the early stages, while the $gBest$ topology satisfies the requirements of tuning the search areas in the late stages. If we can dynamically adapt the topologies, it will provide an effective way to balance between exploration and exploitation in the entire convergence period. This paper presents a dynamic neighborhood topology through gradually increasing the number of connections for each particle in the population. We have formalised this topology using random graph representations. In order to validate our proposed method, we have tested the PSO on 31 benchmark test functions. The results show that the changes in the PSO result on better performance than a number of the state of the art algorithms on almost all of the functions.

The rest of this paper is organised as follows. In Section 2, we will give a detailed review on the related work of this paper. The proposed PSO-GIDN is presented in Section 3. In Section 4 we will provide our experimental results and analysis. Finally in Section 5 we will briefly summarise the contributions of this paper.

2. Background Research

In this section, we first introduce the PSO, and then review the research on the neighbourhood topologies.

2.1. Particle Swarm Optimization

In the PSO, there is a population of solutions referred to as particles. Particles fly around the d-dimensional solution space, and are evaluated according to a fitness criteria after each iteration. The i-th particle's position is represented by a vector $\overrightarrow{x}_{i,t} = (x_{i1}, x_{i2}, \ldots, x_{id})$ (where t is the iteration counter). The flying velocity for a particle i is represented by a vector $\overrightarrow{v}_{i,t} = (v_{i1}, v_{i2}, \ldots, v_{id})$. In every iteration, each particle's flying velocity is updated according to the following two positions. The first one is the position at which its best fitness has achieved so far. This position is a "personal best position" (cognitive component) and is denoted by a vector $\overrightarrow{pb}_{i,t}$. The second position is the best position obtained so far by the particles in its neighbourhood. This position is a "neighbourhood best position" (social component) and is represented by $\overrightarrow{nb}_{i,t}$. Traditionally, the velocity update equation is $\overrightarrow{v}_{i,t} = w * \overrightarrow{v}_{i,t-1} + \overrightarrow{U}[0, c_1] \otimes (\overrightarrow{pb}_{i,t-1} - \overrightarrow{x}_{i,t-1}) + \overrightarrow{U}[0, c_2] \otimes (\overrightarrow{nb}_{i,t-1} - \overrightarrow{x}_{i,t-1})$, where $\overrightarrow{U}[m, n]$ is a vector of random real numbers distributed over $[m, n]$, and w is the inertia weight that has been shown to provide improved performance [7]. There are a number of modified versions of the velocity update equation. Clerc and Kennedy [8] introduced the constriction factor χ making the following equation.

$$\begin{aligned}\overrightarrow{v}_{i,t} = \chi(\overrightarrow{v}_{i,t-1} + \overrightarrow{U}[0, \varphi_1] \otimes (\overrightarrow{pb}_{i,t-1} - \overrightarrow{x}_{i,t-1}) \\ + \overrightarrow{U}[0, \varphi_2] \otimes (\overrightarrow{nb}_{i,t-1} - \overrightarrow{x}_{i,t-1}))\end{aligned} \tag{1}$$

where $\chi = \frac{2}{|2-\varphi-\sqrt{\varphi^2-4\varphi}|}$, $\varphi = \varphi_1 + \varphi_2$ and $\varphi > 4.0$. After updating velocity, each particle updates its position based on its velocity using the following equation.

$$\overrightarrow{x}_{i,t} = \overrightarrow{x}_{i,t-1} + \overrightarrow{v}_{i,t} \tag{2}$$

The PSO using the $lBest$ topology and the Equations (1) and (2) has become the standard PSO [9].

From the discussion above, we can see that the PSO mainly includes two components: the cognitive and social components. The social component plays a key role in the success of the PSO. We usually use the neighborhood topologies to depict the social interactions among particles. There is a body of research on the neighbourhood topologies which we discuss in the followings.

2.2. Neighbourhood Topologies in the PSOs

The neighbourhood topology or sociometric structure indicates the connections between particles. It determines the spread of the information[1]. The information flows faster between connected pairs of individuals while slows down by the presence of the intermediaries. The greater connectivity speeds up convergence, but it does not tend to improve the population's ability to discover global optima. The neighbourhood topology has a strong influence on the particles' search behaviour, and subsequently on the PSOs' success.

In order to find a topology that works for a wide range of problems, various topologies have been proposed and examined. In the following, we first discuss the representation of the topologies, and then review the existing topologies including static and dynamic topologies.

2.2.1. Representation

Neighbourhood topologies in the PSOs are usually defined informally, using only ordinary language or diagrams. It is unambiguous for many simple topologies, but not for more complex and dynamic topologies due to its inherent imprecision.

In order to more accurately represent neighbourhood topologies in the PSOs, Mendes used undirected graphs to represent social topologies [10]. Undirected graphs can only be used to model symmetric relations or bidirectional interactions. In order to model more general relationships, such as single direction communications between particles, we need a model of directed graphs. A directed graph or digraph G is a triple consisting of a vertex set $V(G)$, an edge set $E(G)$, and a function $\Re(G)$ assigning each edge an *ordered pair* of vertices. The first vertex of the ordered pair is the $tail$ of the edge, and the second is the $head$. Each ordered pair or each edge in a digraph is the $(tail, head)$ pair that represents a link from $tail$ to $head$. We can see that the population structure in the PSOs can be easily represented by a digraph G. The vertex set $V(G)$ is all the particles $(p_1, p_2, \ldots, p_n)$. Note that the neighbourhood topologies may change at each iteration in a dynamic topology. In this case, we can use G_t to denote the topology at iteration t.

The neighbourhood of a particle relates to the concepts of *in-neighbourhood set* and *out-neighbourhood set*. The group of the particles that exerts influence over the particle p_i at iteration t is the *in-neighbourhood set* $H_t^+(p_i) = \{x \in$

[1]The information mainly refers to the best positions in the swarm.

$V(G) : x \rightarrow p_i\}$. In other words, the particles from the *in-neighbourhood set* send their personal best positions to the particle p_i. The *out-neighbourhood set* is the group of particles to whom the particle p_i contributes its personal best information. It is formally defined as $H_t^-(p_i) = \{x \in V(G) : p_i \rightarrow x\}$. These two sets are complementary. Therefore we only need to specify one set when we define a topology. In this paper, we prefer to define the *in-neighbourhood set* to construct a topology.

The main advantages of using the digraph representation is that we can explicitly model the information flow direction. Most recent work on the neighbourhood topologies has been adopted this representation [11, 12, 13].

2.2.2. Static Topologies

In the static (or fixed) topologies, the connections do not change over time. The earliest and most common used topologies are the $gBest$ and $lBest$ topologies [14, 7, 9]. In the $gBest$ topology, all particles are fully connected with each other, and the information flows fastest. Subsequently, the PSOs used the $gBest$ converge quickly but are likely to be trapped in local optima. In the $lBest$ topology, each particle shares its information with two adjacent neighbours. Therefore, the flow of information is much slower. The PSOs adopted $lBest$ have stronger ability to explore different regions, but take longer time to converge. Inspired by the idea of "small worlds", Kennedy studied the effects of randomly changed connections of a number of social networks including $gBest$ and $lBest$ [15]. The test results show that the neighbourhood topologies can significantly affect the PSO's performance, and the effects are also dependent on the test functions. Recently, Kennedy and Mendes have systematically examined 70 different topologies [6, 10]. These topologies not only include the traditional regular topology such as von Neumann, pyramid and $lBest$ topologies, but also a large of random graphs[2] with varying degrees of separation or levels of clustering. They find that the von Neumann topology performs pretty well among these topologies, but their research has not precisely identified the topology factors that lead to best performance on a range of problems. All the topologies mentioned above are only considered mutual communications between connected pairs of particles. More recently, Muñoz-Zavala et al. have proposed a new neighbourhood structure, called "singly-linked ring". There is no mutual interactions between adjacent particles [13]. In this topology a particle k only communicates with par-

[2]These random graphs do not change during a trial, and therefore they are also static topologies.

ticles $k-2$ and $k+1$ as neighbors (not $k-1$ and $k+1$ as in the original ring structure). The information in this topology flows even slower than that in the ring topology. Their simulation results show its superiority over the ring and von Neumann topologies in a small range of problems. These studies above are based on the standard PSOs that each particle is influenced by the best performer among its neighbours. Mendes et al. have proposed a new variant of the PSO, called fully informed particle swarm (FIPS) [16]. The FIPS changes the way of processing the information. Each particle is influenced by all its neighbours and, thus, the particles are "fully informed". The FIPS still faces the same problem of finding a neighbourhood topology that works well on a wide rang of problems. There are already a number of research on this problem [17, 18].

In summary, researchers still do not find a static topology that performs effectively for a wide range of problems. Recently, randomized topologies and dynamic topologies have been gained more attention.

2.2.3. Dynamic Topologies

In the *dynamic* topologies, the connections between particles may change over iterations. A number of techniques have been used to manipulate the neighbourhood topologies such as clustering, randomly adding, removing, or migrating edges, and reconstructing neighbourhood periodically.

Suganthan [5] has designed a neighbourhood operator in order to balance the exploration and exploitation. For each particle, a certain percentage of particles close to it were considered as its neighbors. In the early stage, each particle only has a small number of neighbours, while near end of the algorithm, each individual's neighbourhood consists of the entire population. Specifically, a particle p_i's neighbourhood is determined by the distance rate between p_i and any other particle in the population, and a threshold fraction. The threshold fraction is defined: $fra = (3.0t + 0.6max\_t)/max_t$, where t is the current iteration, max_t is the maximum iteration number. If $fra > 0.9$, then the p_i's neighbours are the entire population, otherwise, it is taken to consist of all particles whose distance d from p_i satisfies $d/max\_d < fra$, where max_d is the maximum distance between p_i and any other particle in the population.

Kennedy [19] has used a clustering technique ($k-means$) to reconstruct the populations into several groups at each iteration. The particles in one group or cluster are very close in search space. The effects of using cluster centers as an alternative to using an individual's previous best p_b or its neighborhood previous best n_b are studied. The preliminary study demonstrates that particle swarm search is relatively effective when individuals are influenced by the centers

of their own clusters, and is not generally good when they are influenced by the neighbors' cluster centers. However, the clustering using $k - means$ adds some extra computational cost. Liang and Suganthan [20] have introduced a dynamic multi-swarm PSO (PSO-DMS). In the PSO-DMS, the population is divided into a number of swarms randomly, and the particles in each swarm are organised using the $lBest$ topology. These swarms are regrouped frequently in order to exchange information between swarms.

Mohais et al [11, 21] have proposed to use random neighbourhoods in the PSOs, together with dynamism operators. Their random neighbourhoods can be represented using directed graphs as the relationships between particles are single directional. Both the size and member of the *in-neighbourhood set* $H_t^+(p_i)$ are generated uniformly. Two methods of dynamism called random edge migration and total re-structuring are given in [11, 21].

A number of other dynamic topologies have also been examined such as the scale-free characteristics topologies and self-adjusting neighbourhoods [22, 23].

In summary, the neighbourhood topologies have become an active research direction. These existing studies on the various topologies have gained insight into the effects of the topologies on the performance of the PSOs. However, these studies have the following common disadvantages.

- Only a small number of test functions (usually 6 functions) are used to validate the proposed topologies, and consequently the effectiveness of the topologies is not fully examined.
- Much extra computation cost is added for some complex dynamic topologies, which makes the PSOs run much slower.

In this paper, we present a new dynamic topology in order to balance between exploration and exploitation of search space. To address the disadvantages mentioned above, we validate this topology on a wide range of test functions. Furthermore, our topology requires very little extra computation cost compared with the original PSOs.

3. PSO with Gradually Increasing Directed Neighbourhoods

We aim to provide a neighbourhood topology that can effectively balance between the exploration and exploitation of the solution space. In the early stage of the iteration, the PSOs should focus on exploring the whole parameter space in

order to find a promising search space, and in the late stage all the particles should work together to exploit the best area found so far.

We design a new neighbourhood topology, called "Gradually Increasing Directed Neighbourhoods (GIDN)", to satisfy the desired balance between exploration and exploitation in the entire stage. Specifically, at the beginning each particle only communicates with a small number of particles. This forms many small swarms in the population and thus, improves the exploration ability in the early stage of the evolution. The neighbourhood of each particle increases over time, and each particle is connected with more individuals. In the late stage, all particles will be connected with each other, and share all the information together which improves the exploitation ability.

In our model, we gradually add connections between particles and these connections between particles are randomly chosen and also have directions. So we choose a random directed graph $G(N, b, \gamma, t)$ to formally define our GIDN. We define the *in-neighbourhood set* $H_t^+(p_i)$ for any particle (p_i) in the $G(N, b, \gamma, t)$ first. The size of $H_t^+(p_i)$ at iteration t is determined by the following equation.

$$|H_t^+(p_i)| = \lfloor (\frac{t}{max_t})^\gamma * N + b \rfloor \tag{3}$$

where $\lfloor x \rfloor$ is the largest integer not greater than x (the floor function), max_t is the maximum iteration number, N denotes the size of the population, b and γ are two parameters. The parameter b is the number of neighbours that each particle begins with. We suggest that b is set to a small number such as 2 or 3 in order to create lots of small swarms in the population. The parameter γ controls the neighbourhood size increasing speed and subsequently the information flow speed. Thus, γ can be used to control exploration and exploitation. $f(\gamma) = (\frac{t}{max_t})^\gamma$ is a decreasing function because of $0 < \frac{t}{max_t} \leq 1$. Our results show that $\gamma = 2$ provides a better balance between exploration and exploitation.

From Equation (3), we observe that each vertex (or particle) starts with b edges and at iteration t adds $|H_t^+(p_i)| - |H_{t-1}^+(p_i)|$ new neighbours or edges. How to choose new neighbours for p_i? In this paper, we investigate three different strategies. The first one is to choose particles randomly as their new neighbours. Another one is to choose the particles that are nearest in the search space. The last one is to choose nearest particles in the function space. Our results show that these strategies have no significant effects on the PSO's performance. Therefore, we recommend to choose randomly from the population as it adds the least computational cost.

Algorithm 1: The PSO with Gradually Increasing Directed Neighbourhoods (PSO-GIDN)

Step 1 : Initialisation: randomly generate each particle's position and velocity; Set each particle's neighbour number ($|H^+_{t=0}(p_i)| = 0$);

Step 2 : Renew each particle's neighbourhood. Firstly, obtain each particle's neighbour number ($|H^+_t(p_i)|$) using Equation (3), then update the topology as follows:

for $i \leftarrow 1$ **to** N **do**

  if $|H^+_t(p_i)| > |H^+_{t-1}(p_i)|$ **then**

    Randomly choose $|H^+_t(p_i)| - |H^+_{t-1}(p_i)|$ distinct particles that still do not have connections with the particle p_i as p_i's new neighbours;

Step 3 : Evaluate: update each particle's fitness according to the fitness function;

Step 4 : Update: 1) If the current position is better than $\overrightarrow{pb}_{i,t}$, then update $\overrightarrow{pb}_{i,t}$. 2) If this is a better position than $\overrightarrow{nb}_{i,t}$ in its neighbourhood, then update $\overrightarrow{nb}_{i,t}$;

Step 5 : Update each particle's position and velocity according to Equation (1) and (2);

Step 6 : Check stop criterion: If not, return to Step 2 , otherwise output the best solution found so far.

Algorithm 1 shows our proposed algorithm (PSO-GIDN). The *step 2* in the PSO-GIDN indicates the process of the GIDN.

4. Experimental Results

In this section, we first examine the parameter settings for the PSO-GIDN, and then evaluate its performance through comparing with a number of the existing PSOs.

Table 1 shows all the test functions used in our experiments. The first 6 functions ($f_1 \sim f_6$) are some standard test problems which have been widely used to validate new algorithms [20, 17]. The rest of 25 functions ($f_7 \sim f_{31}$) are proposed by Suganthan et al [24]. These 25 functions are constructed from some basic test functions through adding noise, shifting, rotating or hybrid composition, etc. Due to these adjustments, these functions become more challenging to optimise.

Table 1: Test Functions (Uni.=Unimodal, Multi.=Multimodal, Sh.=Shifted, SR=Shifted and Rotated, HC=Hybrid Composition, RHC=Rotated and HC, GB=Global on Bounds, NC=Non-Continuous, and NM=Number Matrix)

No.	Type	Description	Bounds	Optimum
f_1	Uni.	Sphere	[-5.12, 5.12]	0.0
f_2	Uni.	Rosenbrock	[-2.048, 2.048]	0.0
f_3	Multi.	Ackley	[-30, 30]	0.0
f_4	Multi.	Griewank	[-600, 600]	0.0
f_5	Multi.	Rastrigin	[-5.12, 5.12]	0.0
f_6	Multi.	Schaffer	[-100, 100]	0.0
f_7	Uni.	Sh. Sphere	[-100, 100]	-450
f_8	Uni.	Sh. Schwefel 1.2	[-100, 100]	-450
f_9	Uni.	SR Elliptic	[-100, 100]	-450
f_{10}	Uni.	f_8 with noise	[-100, 100]	-450
f_{11}	Uni.	Schewefel 2.6 GB	[-100, 100]	-310
f_{12}	Multi.	Sh. Rosenbrock	[-100, 100]	390
f_{13}	Multi.	SR Griewank	[0, 600]	-180
f_{14}	Multi.	SR Ackley GB	[-32, 32]	-140
f_{15}	Multi.	Sh. Rastrigin	[-5, 5]	-330
f_{16}	Multi.	SR Rastrigin	[-5, 5]	-330
f_{17}	Multi.	SR Weierstrass	[-0.5, 0.5]	90
f_{18}	Multi.	Schwefel 2.13	[$-\pi$, π]	-460
f_{19}	Multi.	Sh. Expanded F8F2	[-3, 1]	-130
f_{20}	Multi.	SR Scaffer F6	[-100, 100]	-300
f_{21}	Hybrid	HC Function	[-5, 5]	120
f_{22}	Hybrid	RHC Function 1	[-5, 5]	120
f_{23}	Hybrid	f_{22} with noise	[-5, 5]	120
f_{24}	Hybrid	RHC Function 2	[-5, 5]	10
f_{25}	Hybrid	f_{24} with basin	[-5, 5]	10
f_{26}	Hybrid	f_{24} with GB	[-5, 5]	10
f_{27}	Hybrid	RHC function 3	[-5, 5]	360
f_{28}	Hybrid	f_{27} with NM	[-5, 5]	360
f_{29}	Hybrid	NC Rotated f_{27}	[-5, 5]	360
f_{30}	Hybrid	RHC function 4	[-5, 5]	260
f_{31}	Hybrid	f_{30} without bounds		260

We use the following parameter settings for all the experiments unless otherwise specified. The population size (N) is set to 60. The parameter χ is set to 0.72984, both φ_1 and φ_2 are set to 2.05, and b is set to 3. For the functions $f_1 \sim f_6$, the iteration number is 1000, while for $f_7 \sim f_{31}$ it is set to 5000. The dimensions for all the functions are set to 30. Each set of the results are from 25 independent runs. We conduct all the experiments on an Intel (R) CPU T8300, 2.40 GHz, 4 GB RAM and Windows 7 OS computer using Java language.

4.1. Parameter Setting Analysis for PSO-GIDN

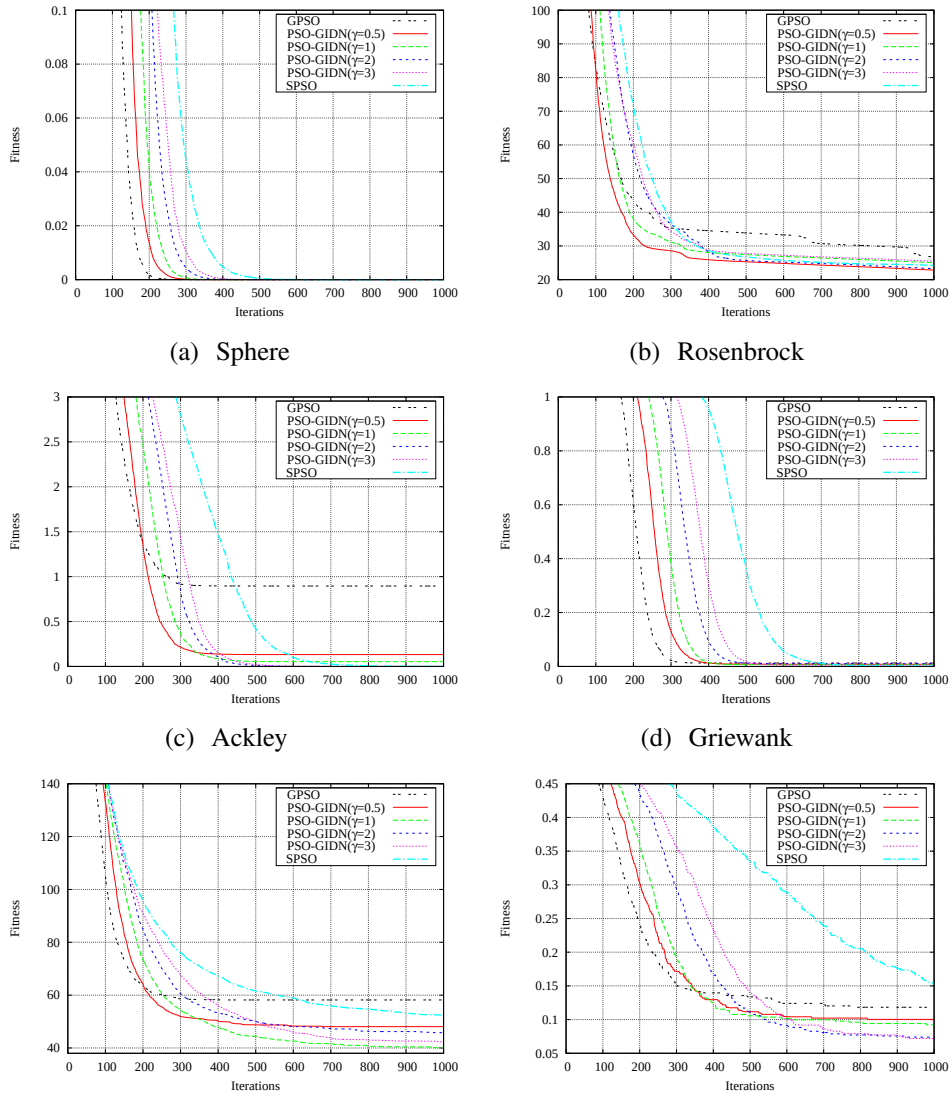
The first 6 functions ($f_1 \sim f_6$) are used to analyse the parameter settings in the PSO-GIDN. The main concern for the PSO-GIDN is the parameter γ setting which determines the speed of information flow among particles. While another interesting issue is that how to choose their neighbours when the particles dynamically increase their neighbours. These two parameters are examined in the following.

4.1.1. γ

As discussed earlier, γ determines the evolution of population topologies in the PSO-GIDN. In order to find a proper value for γ, we examine the convergence of the PSO-GIDN under a group of settings. Specifically, γ is set to 0.5, 1, 2 and 3. The particles randomly choose their new neighbours. From Equation (3), we can get: $N > c_k(\gamma = 0.5) > c_k(\gamma = 1) > c_k(\gamma = 2) > c_k(\gamma = 3) > 2$. Therefore, at early stage of the evolution, the convergence rate should be: GPSO $>$ PSO-GIDN $(\gamma = 0.5)$ $>$ PSO-GIDN $(\gamma = 1)$ $>$ PSO-GIDN $(\gamma = 2)$ $>$ PSO-GIDN $(\gamma = 3)$ $>$ SPSO (the GPSO and SPSO represents the PSO with $gBest$ and $lBest$ topologies respectively). Our experimental results confirm this. However, we are more interested in the performance in the late stage. Therefore, in Figure 1, we only show the convergence curves in the late stage. From these curves, we observe that no single γ value can always perform better on all the 6 functions. For example, PSO-GIDN $(\gamma = 0.5)$ performs better on two unimodal functions (Sphere and Rosenbrock), while PSO-GIDN $(\gamma = 1)$ performs better on Rastrigin. However, PSO-GIDN $(\gamma = 2)$ generally performs better on most of the multimodal functions and also performs well in the unimodal functions. Accordingly, we recommend that γ is set to 2.

4.1.2. How to choose a neighbour?

In the PSO-GIDN, each particle dynamically increases the size of their neighbourhood. It might be interesting to investigate the impacts of the choosing neighbour strategies on the performance of the PSO-GIDN. We examine three different



(a) Sphere (b) Rosenbrock

(c) Ackley (d) Griewank

(e) Rastrigin (f) Schaffer

Figure 1: Convergence

strategies. The first one is to choose particles randomly as their new neighbours. Another one is to choose the particles that are nearest in the search space. The last one is to choose nearest particles in the function space.

Table 2 shows the results obtained from different choosing strategies. Here, γ is fixed to 2. The nearest neighbours in the search space strategy has better

Table 2: PSO-GIDN Performance with Different Choosing Neighbour Strategies

No.	Random neighbours		Nearest neighbours in Search Space		Nearest neighbours in Function Space	
	Mean (STD)	Time (S)	Mean (STD)	Time (S)	Mean (STD)	Time (S)
f_1	2.85E-16 (2.61E-16)	0.739	**2.38E-16 (3.57E-16)**	3.073	1.42E-15 (2.81E-15)	0.981
f_2	2.32E1 (2.57E0)	0.937	**2.24E1 (3.62E0)**	3.296	2.51E1 (1.1E1)	1.217
f_3	1.06E-7 (9.68E-8)	0.905	**9.39E-8 (5.6E-8)**	3.184	1.67E-7 (1.22E-7)	1.088
f_4	**6.99E-3 (1.02E-2)**	0.937	1.04E-2 (1.06E-2)	3.174	7.49E-3 (7.08E-3)	1.046
f_5	**4.52E1 (1.16E1)**	0.958	4.76E1 (1.14E1)	3.163	6.76E1 (1.63E1)	1.051
f_6	**7.35E-2 (1.85E-2)**	0.757	8.08E-2 (2.03E-2)	2.959	7.59E-2 (2.48E-2)	0.860

Table 3: PSO Algorithms Used in the Comparisons

Algorithm	Topologies	Ref.
SPSO	$lBest$	[9]
GPSO	$gBest$	[7]
VPSO	von Neumann	[6]
PSO-NO	Neighbourhood Operator	[5]
PSO-RDN	Randomized Directed Neighbourhood	[11]
PSO-DMS	Dynamic Multi-swarm	[20]

performance on first three functions, however, it consumes much more time that the other two strategies. While the randomly choosing neighbour strategy works better on the last three functions that are more difficult to optimise, and it also performs well on the first three functions. Furthermore, this strategy consumes less time than the other two strategies. Based on these results, we suggest choosing each particle's new neighbours randomly.

4.2. Comparisons with Other PSO algorithms

To validate the proposed PSO-GIDN, we compare the PSO-GIDN with a number of existing PSO algorithms on the 31 functions. We select the existing PSO algorithms as comparisons based on neighbourhood topologies. These PSO algorithms are shown in Table 3. The neighbourhood topologies chosen represent the state of the art topologies. The PSO with the ring topology is known as the standard PSO (SPSO) [9]. The PSO with the $gBest$ topology and von Neumann (or

Table 4: Comparisons between PSO-GIDN and other PSO Algorithms

No.	PSO-GIDN	SPSO	GPSO	VPSO	PSO-NO	PSO-RDN	PSO-DMS
	Mean (STD)	Mean (STD)	Mean (STD)	Mean (STD)	Mean (STD)	Mean (STD)	Mean (STD)
f_1	2.85E-16 (2.61E-16)	6.41E-9 (3.73E-9)	**1.88E-21 (3.85E-21)**	2.41E-12 (1.89E-12)	2.96E-21 (5.8E-21)	1.01E-10 (8.46E-11)	1.35E-8 (7.12E-9)
f_2	2.32E1 (2.57E0)	2.39E1 (2.67E0)	3.35E1 (2.38E1)	2.46E1 (1.61E0)	**2.26E1 (1.9E0)**	2.73E1 (1.06E1)	2.53E1 (1.2E0)
f_3	**1.06E-7 (9.68E-8)**	7.03E-4 (3E-4)	1.11E0 (8.47E-1)	8.64E-6 (4.74E-6)	8.26E-1 (7.91E-1)	5.02E-5 (1.91E-5)	7.08E-4 (2.44E-4)
f_4	6.99E-3 (1.02E-2)	1.69E-3 (3.73E-3)	1.01E-2 (1.05E-2)	4.56E-3 (8.13E-3)	3.24E-2 (7.17E-2)	7.19E-3 (8.06E-3)	**7.52E-4 (2.11E-3)**
f_5	**4.52E1 (1.16E1)**	4.97E1 (8.76E0)	5.34E1 (1.35E1)	4.54E1 (7.93E0)	4.94E1 (9.32E0)	4.99E1 (1.66E1)	9.37E1 (1.1E1)
f_6	7.35E-2 (1.85E-2)	1.45E-1 (3.16E-2)	1.18E-1 (4.23E-2)	7.82E-2 (1.33E-2)	1.12E-1 (3.27E-2)	**6.34E-2 (1.97E-2)**	8.9E-2 (2.18E-2)
f_7	**-358.98 (91.49)**	542.3 (507.22)	-103.29 (394.69)	-52.41 (278.48)	36.19 (561.05)	-58.65 (365.46)	2397.59 (540.74)
f_8	**-375.68 (255.46)**	30.33 (725.02)	1386.51 (2838.2)	-100.6 (562.31)	1759.26 (3201.93)	984.66 (1531.7)	4784.2 (1837.98)
f_9	**6267643 (8517300)**	7710409 (6294142)	48064757 (45618893)	6546530 (7183801)	14596032 (15094962)	11951976 (9738194)	37355691 (14378455)
f_{10}	**1745.86 (1424.34)**	10967.21 (3155.72)	5909.19 (6819.21)	3283.85 (1518.82)	4203.11 (4526.62)	4106.26 (1663.76)	12454.32 (2225.33)
f_{11}	8486.32 (1056.27)	10620.86 (1555.64)	12152.89 (2378.88)	**8285.75 (984.11)**	10733.54 (2195.99)	8595.76 (1379.38)	11839.66 (1299.09)
f_{12}	**1764509 (2554673)**	26285911 (18942482)	7030737 (6664173)	3856860 (3088324)	4920850 (4995025)	5105230 (4205608)	88985497 (39786538)
f_{13}	**4847.81 (48.98)**	4944.71 (50.8)	4969.01 (96.73)	4858.35 (32.47)	4933.28 (78.29)	4888.9 (38.34)	5101.59 (40.22)
f_{14}	-119.12 (0.06)	-119.11 (0.04)	-119.07 (0.05)	-119.11 (0.06)	-119.1 (0.06)	**-119.19 (0.03)**	-119.06 (0.05)
f_{15}	-265.21 (14.53)	-250.15 (6.94)	-255.0 (20.48)	-264.38 (12.98)	-263.38 (19.06)	**-278.94 (11.76)**	-247.23 (14.02)
f_{16}	-222.16 (23.28)	-201.26 (32.81)	-191.69 (39.34)	**-244.66 (18.05)**	-215.78 (35.43)	-186.09 (38.94)	-161.09 (33.75)
f_{17}	117.44 (2.7)	117.49 (3.17)	117.22 (4.63)	117.45 (3.16)	119.35 (2.29)	118.47 (4.52)	**113.43 (3.05)**
f_{18}	9338.53 (6689.24)	**9231.07 (7488.09)**	22194.72 (18354.1)	15727.59 (11808.76)	20742.44 (22648.59)	24464.86 (18775.45)	23403.82 (9472.82)
f_{19}	-126.14 (0.98)	-125.26 (0.92)	-125.73 (1.25)	**-126.31 (0.94)**	-126.03 (0.85)	-125.42 (1.38)	-122.01 (1.56)
f_{20}	-287.56 (0.4)	-287.65 (0.22)	-287.61 (0.48)	**-287.91 (0.38)**	-287.79 (0.63)	-287.57 (0.32)	-287.39 (0.34)
f_{21}	450.8 (140.63)	**421.34 (95.91)**	614.64 (146.66)	540.22 (129.92)	568.5 (144.24)	500.92 (144.62)	499.04 (93.54)
f_{22}	251.6 (27.08)	269.05 (25.23)	393.05 (144.12)	**240.93 (22.66)**	281.8 (53.86)	291.04 (40.42)	324.37 (24.63)
f_{23}	273.2 (38.58)	354.56 (22.64)	479.05 (201.08)	**271.16 (29.58)**	339.57 (95.16)	342.53 (34.31)	390.71 (24.15)
f_{24}	**998.33 (15.52)**	1042.03 (11.97)	1049.27 (24.4)	1011.26 (10.85)	1039.65 (25.78)	1015.41 (8.79)	1043.49 (13.42)
f_{25}	**1004.86 (15.76)**	1036.68 (12.96)	1044.82 (20.96)	1014.09 (11.01)	1025.61 (20.36)	1010.55 (9.43)	1047.03 (14.16)
f_{26}	**1007.03 (11.13)**	1029.48 (14.2)	1069.18 (31.25)	1015.65 (12.48)	1025.53 (21.87)	1009.99 (10.55)	1041.28 (15.07)
f_{27}	**1042.45 (227.39)**	1336.66 (155.14)	1420.82 (230.38)	1087.22 (166.98)	1229.7 (274.19)	1196.23 (212.13)	1438.75 (106.17)
f_{28}	1394.24 (30.07)	1422.47 (27.7)	1468.69 (45.48)	**1387.45 (17)**	1441.79 (38.98)	1394.22 (27.61)	1442.71 (23.7)
f_{29}	**1024.04 (110.06)**	1346.62 (139.12)	1403.86 (202.68)	1139.15 (210.43)	1394.97 (224.12)	1206.37 (208.47)	1469.97 (64.86)
f_{30}	**463.61 (4.87)**	780.89 (180.16)	773.84 (407.61)	505.16 (28.43)	527.06 (202.34)	502.13 (57.36)	1058.28 (133.27)
f_{31}	2080.97 (223.59)	**2064.28 (178.45)**	2216.93 (173.24)	2236.79 (120.2)	2091.93 (171.21)	2157.19 (183.79)	2229.2 (126.48)
Summary							
Better		26	28	23	28	26	29
Worse		5	3	8	3	5	2

Square) topology are denoted as GPSO and VPSO respectively. The PSO-NO is the PSO with a neighbourhood operator that was proposed by Suganthan [5]. The PSO-RDN is the PSO with randomised directed neighbourhoods and edge migrations [11]. The dynamic multi-swarm PSO (PSO-DMS) uses many small swarms and regroups these swarms frequently [20]. In these 6 topologies, $lBest$, $gBest$ and von Neumann topologies are static while the rest are dynamic topologies. In order to compare the effects of neighbourhood topologies purely, all these PSO algorithms employ Equations (1) and (2) to update each particle's velocity and

Table 5: T-test Results

No.	SPSO	GPSO	VPSO	PSO-NO	PSO-RDN	PSO-DMS
	T-value (P-value)	T-value (P-value)	Mean (STD)	Mean (STD)	Mean (STD)	Mean (STD)
f_1	-8.592 (0.000)	5.460 (0.000)	-6.375 (0.000)	5.460 (0.000)	-5.969 (0.000)	-9.480 (0.000)
f_2	-0.944 (0.350)	-2.151 (0.037)	-2.308 (0.025)	0.939 (0.353)	-1.880 (0.066)	-3.702 (0.001)
f_3	-11.715 (0.000)	-6.553 (0.000)	-9.000 (0.000)	-5.221 (0.000)	-13.113 (0.000)	-14.506 (0.000)
f_4	2.440 (0.018)	-1.062 (0.293)	0.931 (0.356)	-1.754 (0.086)	-0.077 (0.939)	2.994 (0.004)
f_5	-1.548 (0.128)	-2.303 (0.026)	-0.071 (0.944)	-1.411 (0.165)	-1.160 (0.252)	-15.169 (0.000)
f_6	-9.763 (0.000)	-4.819 (0.000)	-1.031 (0.308)	-5.124 (0.000)	1.869 (0.068)	-2.711 (0.009)
f_7	-8.743 (0.000)	-3.155 (0.003)	-5.229 (0.000)	-3.476 (0.001)	-3.986 (0.000)	-25.132 (0.000)
f_8	-2.641 (0.011)	-3.092 (0.003)	-2.227 (0.031)	-3.323 (0.002)	-4.380 (0.000)	-13.903 (0.000)
f_9	-0.681 (0.499)	-4.503 (0.000)	-0.125 (0.901)	-2.403 (0.020)	-2.197 (0.033)	-9.301 (0.000)
f_{10}	-13.317 (0.000)	-2.988 (0.004)	-3.693 (0.001)	-2.589 (0.013)	-5.389 (0.000)	-20.265 (0.000)
f_{11}	-5.676 (0.000)	-7.043 (0.000)	0.695 (0.491)	-4.611 (0.000)	-0.315 (0.754)	-10.014 (0.000)
f_{12}	-6.415 (0.000)	-3.689 (0.001)	-2.610 (0.012)	-2.813 (0.007)	-3.395 (0.001)	-10.939 (0.000)
f_{13}	-6.866 (0.000)	-5.589 (0.000)	-0.897 (0.374)	-4.628 (0.000)	-3.303 (0.002)	-20.021 (0.000)
f_{14}	-0.693 (0.491)	-3.201 (0.002)	-0.589 (0.558)	-1.179 (0.244)	5.217 (0.000)	-3.841 (0.000)
f_{15}	-4.676 (0.000)	-2.033 (0.048)	-0.213 (0.832)	-0.382 (0.704)	3.673 (0.001)	-4.452 (0.000)
f_{16}	-2.598 (0.012)	-3.333 (0.002)	3.819 (0.000)	-0.752 (0.455)	-3.975 (0.000)	-7.448 (0.000)
f_{17}	-0.060 (0.952)	0.205 (0.838)	-0.012 (0.990)	-2.697 (0.010)	-0.978 (0.333)	4.922 (0.000)
f_{18}	0.054 (0.958)	-3.291 (0.002)	-2.354 (0.023)	-2.414 (0.020)	-3.795 (0.000)	-6.064 (0.000)
f_{19}	-3.273 (0.002)	-1.291 (0.203)	0.626 (0.534)	-0.424 (0.673)	-2.127 (0.039)	-11.209 (0.000)
f_{20}	0.986 (0.329)	0.400 (0.691)	3.172 (0.003)	1.541 (0.130)	0.098 (0.923)	-1.619 (0.112)
f_{21}	0.865 (0.391)	-4.032 (0.000)	-2.335 (0.024)	-2.921 (0.005)	-1.242 (0.220)	-1.428 (0.160)
f_{22}	-2.357 (0.023)	-4.823 (0.000)	1.511 (0.137)	-2.505 (0.016)	-4.053 (0.000)	-9.940 (0.000)
f_{23}	-9.094 (0.000)	-5.027 (0.000)	0.210 (0.835)	-3.232 (0.002)	-6.714 (0.000)	-12.909 (0.000)
f_{24}	-11.148 (0.000)	-8.808 (0.000)	-3.414 (0.001)	-6.866 (0.000)	-4.788 (0.000)	-11.005 (0.000)
f_{25}	-7.797 (0.000)	-7.619 (0.000)	-2.401 (0.020)	-4.030 (0.000)	-1.549 (0.128)	-9.952 (0.000)
f_{26}	-6.222 (0.000)	-9.368 (0.000)	-2.577 (0.013)	-3.769 (0.000)	-0.965 (0.339)	-9.141 (0.000)
f_{27}	-5.344 (0.000)	-5.844 (0.000)	-0.793 (0.431)	-2.628 (0.011)	-2.473 (0.017)	-7.896 (0.000)
f_{28}	-3.452 (0.001)	-6.828 (0.000)	0.983 (0.331)	-4.829 (0.000)	0.002 (0.998)	-6.330 (0.000)
f_{29}	-9.092 (0.000)	-8.234 (0.000)	-2.424 (0.019)	-7.428 (0.000)	-3.867 (0.000)	-17.453 (0.000)
f_{30}	-8.802 (0.000)	-3.805 (0.000)	-7.203 (0.000)	-1.567 (0.124)	-3.346 (0.002)	-22.296 (0.000)
f_{31}	0.292 (0.772)	-2.403 (0.020)	-3.069 (0.004)	-0.195 (0.847)	-1.317 (0.194)	-2.885 (0.006)
Summary						
Statistically Better	21	26	15	20	17	27
Statistically Same	9	4	14	10	12	2
Statistically Worse	1	1	2	1	2	2

position. This setting is different with the original settings in the PSO-DMS and GPSO. The parameters χ, c_1 and c_2 are set as stated earlier. For the PSO-RDN, the neighbourhood size for each particle is generated randomly between 1 and 4, and the migration rate of 1 per iteration is used. For the PSO-DMS, each swarm has 3 particles, and the regroup period is set to 10. For the PSO-GIDN, the γ is set to 2, and the randomly choosing neighbour strategy is applied.

Table 4 shows the mean value and standard deviation over 25 individual trials

for these PSO algorithms. We find that the PSO-GIDN performs better than the other PSO algorithms on almost of all the 31 functions. Specifically, the PSO-GIDN performs better than the SPSO on 26 functions, better than the GPSO on 28 functions, better than the VPSO on 23 functions, better than the PSO-NO on 28 functions, better than the PSO-RDN on 26 functions, and better than the PSO-DMS on 29 functions. Furthermore, in order to verify whether the performance differences are statistically significant, a t-test was conducted. We use the conventional criteria to determine whether differences are significant. That is if the two tailed p value is less than 5%, the difference is statistically significant, or else, it is not. The t-test results are shown in Table 5. From the summary in Table 5, there are only a very few cases that the PSO-GIDN performs statistically significant worse than the other PSO algorithms. In contrast, the PSO-GIDN performs statistically significant better over the SPSO, GPSO, PSO-NO, PSO-RDN and PSO-DMS on the majority of the functions. Overall the PSO-GIDN also performs better than the VPSO as it performs statistically significant better on 15 functions and statistically same on 14 functions. From these results, we can conclude that the PSO-GIDN provides a better topology and generally performs better than the existing PSO algorithms.

5. Conclusions

Like many other population-based algorithms, the performance of the PSO depends on its ability to balance exploration and exploitation of the search space. In this paper, we aim to improve this trade-off through the neighbourhood structure design. Our study suggests that a productive balance between exploration and exploitation can be achieved by a gradually increasing directed neighbourhood topology. This dynamic topology can support the exploration of promising regions in the search space, while gradually improve the ability of exploitation.

Although our results are promising, future research should continue to test the current topology on more problems and explore other alternative topologies, and we hope that the work we present here represents an important stepping stone to uncovering more nuanced underpinnings between neighbourhood topologies and performance.

[1] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, Vol. 4, IEEE Press, 1995, pp. 1942–1948.

[2] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995, pp. 39–43.

[3] R. Eberhart, Y. Shi, Particle swarm optimization: developments, applications and resources, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2001), Vol. 1, 2001, pp. 81 –86.

[4] M. AlRashidi, M. El-Hawary, A survey of particle swarm optimization applications in electric power systems, Evolutionary Computation, IEEE Transactions on 13 (4) (2009) 913 –918. doi:10.1109/TEVC.2006.880326.

[5] P. Suganthan, Particle swarm optimiser with neighbourhood operator, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999), Vol. 3, 1999, p. 1962 Vol. 3. doi:10.1109/CEC.1999.785514.

[6] J. Kennedy, R. Mendes, Population structure and particle swarm performance, in: Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002), IEEE Computer Society, Washington, DC, USA, 2002, pp. 1671–1676.

[7] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: Proceedings of the IEEE International Conference on Evolutionary Computation (CEC 1998), IEEE Press, Piscataway, NJ, 1998, pp. 69–73.

[8] M. Clerc, J. Kennedy, The particle swarm - explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation 6 (1) (2002) 58–73.

[9] D. Bratton, J. Kennedy, Defining a standard for particle swarm optimization, in: IEEE Swarm Intelligence Symposium, 2007, pp. 120 –127. doi:10.1109/SIS.2007.368035.

[10] R. Mendes, Population topologies and their influence in particle swarm performance, Ph.D. thesis, Escola de Engenharia, Universidade do Minho (2004).

[11] A. Mohais, C. Ward, C. Posthoff, Randomized directed neighborhoods with edge migration in particle swarm optimization, in: Proceedings of the IEEE International Conference on Evolutionary Computation (CEC 2004), Vol. 1, 2004, pp. 548 – 555 Vol.1. doi:10.1109/CEC.2004.1330905.

[12] A. McNabb, M. Gardner, K. Seppi, An exploration of topologies and communication in large particle swarms, in: Proceedings of the Eleventh conference on Congress on Evolutionary Computation (CEC 2009), IEEE Press, Piscataway, NJ, USA, 2009, pp. 712–719.

[13] A. E. Muñoz Zavala, A. Hernández Aguirre, E. R. Villa Diharce, The singly-linked ring topology for the particle swarm optimization algorithm, in: Proceedings of the 11th Annual conference on Genetic and evolutionary computation (GECCO 2009), ACM, New York, NY, USA, 2009, pp. 65–72. doi:http://doi.acm.org/10.1145/1569901.1569911.

[14] R. C. Eberhart, P. K. Simpson, R. W. Dobbins, Computational Intelligence PC Tools, Boston, MA: Academic Press Professional, 1996.

[15] J. Kennedy, Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance, in: Proceedings of the 1999 Congress on Evolutionary Computation, Vol. 3, 1999, pp. 1931–1938.

[16] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, Evolutionary Computation, IEEE Transactions on 8 (3) (2004) 204 – 210.

[17] J. Kennedy, R. Mendes, Neighborhood topologies in fully informed and best-of-neighborhood particle swarms, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 36 (4) (2006) 515 –519.

[18] J. Jordan, S. Helwig, R. Wanka, Social interaction in particle swarm optimization, the ranked fips, and adaptive multi-swarms, in: Proceedings of the 10th annual conference on Genetic and evolutionary computation (GECCO 2008), ACM, New York, NY, USA, 2008, pp. 49–56. doi:http://doi.acm.org/10.1145/1389095.1389103.

[19] J. Kennedy, Stereotyping: improving particle swarm performance with cluster analysis, in: Proceedings of the 2000 Congress on Evolutionary Computation (CEC 2000), Vol. 2, 2000, pp. 1507 –1512 vol.2. doi:10.1109/CEC.2000.870832.

[20] J. Liang, P. Suganthan, Dynamic multi-swarm particle swarm optimizer, in: Proceedings of the 2005 IEEE Swarm Intelligence Symposium, 2005, pp. 124 – 129.

[21] A. Mohais, R. Mendes, C. Ward, C. Posthoff, Neighborhood re-structuring in particle swarm optimization, in: S. Zhang, R. Jarvis (Eds.), AI 2005: Advances in Artificial Intelligence, Vol. 3809 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2005, pp. 776–785.

[22] A. Godoy, F. J. Von Zuben, A complex neighborhood based particle swarm optimization, in: Proceedings of the Eleventh conference on Congress on Evolutionary Computation (CEC 2009), IEEE Press, Piscataway, NJ, USA, 2009, pp. 720–727.

[23] Z. Chen, Z. He, C. Zhang, Particle swarm optimizer with self-adjusting neighborhoods, in: Proceedings of the 12th annual conference on Genetic and evolutionary computation (GECCO-2010), ACM, New York, NY, USA, 2010, pp. 9–14. doi:http://doi.acm.org/10.1145/1830483.1830486.

[24] P. N. Suganthan, N. Hansen, Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization, Tech. rep., Nanyang Technological University, Singapore and KanGAL Report Number 2005005 (2005).