Abstract— Particle Swarm Optimization (PSO) is an optimization method that is inspired by nature and is used frequently nowadays. In this paper we proposed a new dynamic geometric neighborhood based on Voronoi diagram in PSO. Voronoi diagram is a geometric naturalistic method to determine neighbors in a set of particles. It seems that in realistic swarm, particles take Voronoi neighbors into account.

Also a comparison is made between the performance of some traditional methods for choosing neighbors and new dynamic geometric methods like Voronoi and dynamic Euclidean. In this comparison it is found that PSO with geometric neighborhood can achieve better accuracy overall especially when the optimum value is out of the initial range.

Keywords — Particle Swarm Optimization; Voronoi Diagram; Neighborhood Topologies

I. INTRODUCTION

There are many methods for solving various kinds of optimization problems. Some of them are linear and nonlinear programming methods. Some others are evolutionary algorithms like the Genetic Algorithm. Particle Swarm Optimization (PSO) was introduced by Kennedy and Eberhart in 1995 [1]. They proposed their new optimization tool based on swarm theory and evolutionary computation. Since then, a lot of improvements and modifications of PSO have been introduced.

From 1997 to 1999 some works started using neighborhood in PSO [2-5]. They found out that using neighborhood can decrease the probability of falling into local minima. Since by using neighborhood, even if some of the particles fall into local minima, they will only influence some particles, and others can explore the rest of the world.

Suganthan in [5] has used the local best particle instead of the global best, where, the best local particle meant the best particle in the neighborhood of each particle. For each particle, a certain percentage of particles close to it were considered as its neighbors. This percentage can increase linearly. So, at the end of the algorithm the local best can be equivalent to the global best.

In [6], Kennedy and Mendes have discussed some structures of population. They involved a kind of neighborhood called social, which is known as lbest methodology. In this structure, at the initialization point, each particle is associated with some random particles. They also compared traditional gbest (Fig. 1-a) topology to some lbest topologies like von Neumann, star, ring (mostly known as lbest, Fig. 1-b) and pyramid.

Kennedy and Mendes [7] have suggested a new methodology for involving neighborhood in PSO, called Fully-Informed Particle Swarm (FIPS), which uses some portion of each neighbor’s findings instead of the best neighbor and the best experience of the particle. They have indicated that the individual’s experience tends to be overwhelmed by social influence. They also in [8] have announced another version of FIPS algorithm and made some comparison between these methods and social structures. They have reported some good results obtained by Von Neumann topology.

Researchers who have suggested methods that use static neighborhood (i.e. neighbors are fixed during the run), discussed that by this kind of neighborhood, a society is constructed between particles. Once a particle finds a good result, it reports its location to its friends, and by some iterations all of the particles know something about good locations and try to move to them.

Finally in [9], the authors have announced two forms of modifying the neighborhood structure (but preserving the topology) during the run. One of them randomly changes an edge in the graph with some probability (migration) and the other one restructures the whole graph. They reported satisfactory results using the second method. They also concluded that the dynamic neighborhood has better performance than the static one.
It is cited in [10] (page 111) from Reynolds that realistic birds flock could be programmed by implementing three simple rules: match your neighbors' velocity, steer towards the perceived center of the flock, and avoid collisions. It is clear that the term neighbor in this sentence is geometric, rather than social neighbor.

In this study, we will use Voronoi diagram, which supports geometric dynamic neighborhood. Voronoi diagram has many natural representations.

There are some works which combine the Voronoi diagram and PSO. In [11], the authors have used a mechanism called Centroidal Voronoi Tessellation (CVT) for initializing the PSO. This algorithm spreads the particle in the environment and is used for distributing the particles uniformly. In [12], the authors have introduced a novel optimization tool based on the PSO called Quantum Particle Swarm Optimization which uses CVT to distribute numerous quantum particles uniformly to ensure full coverage of the search space. The authors in [13] have used the PSO for solving the path planning problem. They used Voronoi diagram as the initialization of the paths as particles. However, no one has used the Voronoi neighborhood to improve the PSO.

This paper is organized as follows: in section 2 we will first define the Voronoi Diagram, Voronoi neighborhood and next the PSO in detail, and we will focus on using neighborhood in PSO. In section 3 the motivation for using Voronoi neighborhood in PSO and the details of the algorithm will be proposed. Section 4 contains the empirical results and finally, some conclusions will be drawn in section 5.

II. DEFINITIONS AND PRELIMINARY

A. Voronoi Diagram

Consider $S=\{s_1, s_2, \ldots, s_n\}$ as a set of distinct sites in $\mathbb{R}^d$ and $d(s_i, s_j)$ as the Euclidean distance between two sites, $s_i$ and $s_j$. We define the Voronoi region of a site as follows.

**Definition 1.** [14] The Voronoi region of a site, $V(s_i)$, consists of all the points whose nearest site is $s_i$: 

$$V(s_i) = \{x : d(s_i, x) \leq d(s_j, x), \forall j \neq i\} \quad (1)$$

The union of the Voronoi regions of all sites represents a tessellation of the space that is called Voronoi tessellation:

$$V(S) = \bigcup_{i=1}^{n} V(s_i) \quad (2)$$

Note that some points do not have a unique nearest site. The set of all points that have more than one nearest site forms the Voronoi diagram.

**Definition 2.** We also define the adjacency of a site based on Voronoi Diagram. The $adj(s_i)$ consists of all the other sites that are adjacent to $s_i$ in Voronoi tessellation, which means $s_i$ is in $adj(s_i)$ if and only if there is a joint edge between $V(s_i)$ and $V(s_j)$ in Voronoi diagram.

Fig.2 shows an example of Voronoi diagram in two-dimensional space. The points show sites and the edges represent Voronoi diagram. For example in this figure you can see $adj(s_9) = \{s_1, s_2, s_4, s_5, s_6, s_7\}$.

The Voronoi diagram is a fabulous structure that has been observed many times in nature, for example the studies show that galaxies in the universe are not distributed randomly. There are some galactic poles and galaxies are clustered in these poles due to gravity. These clusters form an structure which is like Voronoi diagram [15,16,17].

The Voronoi diagram is also studied in too many diverse sciences namely Anthropology and Archeology, Astronomy, Biology and Ecology, Crystallography and Chemistry, Geography, Zoology, mathematics, pattern recognition, etc owing to its properties [14,18,19,20].

B. Particle Swarm Optimization

Particle Swarm Optimization (PSO) was introduced in 1995 by James Kennedy and Russell Eberhart [1]. It is inspired by the swarming behavior of animals and human social behavior, and can play an important role in optimization procedures. PSO is really similar to the Genetic Algorithm (GA) in the sense that they are evolutionary heuristics and population-based search methods [2].

PSO algorithm is a zero-order, non-calculus-based method (no gradients are needed) that can solve discontinuous, multimodal and non-convex problems. Furthermore, it includes some probabilistic features in the motion of particles. Also PSO does not need genetic operators. So it is easy to implement, needs fewer parameters, and has rapid convergence [21].

At first, particle positions and velocity vectors are generated randomly in space. Each particle adjusts its position and velocity along each dimension, based on the best position that it has encountered so far and the best particle in the swarm. Particles continue this until they find the best position.

In PSO algorithm, the search space is $n$-dimensional, so the $i^{th}$ particle can be represented by an $n$-dimensional vector $X_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T$, and velocity $V_i = [v_{i1}, v_{i2}, \ldots, v_{in}]^T$, where $i = 1, 2, \ldots, N$ and $N$ is the size of the population. The best position encountered by

---

1. Identify the parts of a region under the influence of different Neolithic clans, chiefdoms, ceremonial centers, or hill forts
2. Identify clusters of stars and clusters of galaxies
3. Model and analyze plant competition
4. Study chemical properties of metallic sodium
5. Analyzing patterns of urban settlements
6. Model and analyze the territories of animals
7. Study of positive definite quadratic forms
particle $i$ so far is represented by $P_i$, and the best particle in the swarm is represented by $G$.

In standard PSO algorithm, each particle $i$ calculates its velocity and position in the next iteration $t+1$ with respect to equations (3) and (4):

$$V_i(t+1) = \omega(t)V_i(t) + c_1r_1(P_i(t) - X_i(t)) + c_2r_2(G(t) - X_i(t))$$

(3)

$$X_i(t+1) = X_i(t) + \chi V_i(t+1)$$

(4)

where $\omega(t)$ is the inertia coefficient which gradually decreases linearly from 0.95 in the first iteration to 0.4 and $\chi$ is the constriction factor which is used to limit velocity. In our experiments, it is set to 1 and has no effect.

In the neighborhood version of PSO, equation (3) is changed to equation (5), where $L_i$ is the best particle in the neighborhood of particle $i$.

$$V_i(t+1) = \omega(t)V_i(t) + c_1r_1(P_i(t) - X_i(t)) + c_2r_2(L_i(t) - X_i(t))$$

(5)

In our experiments $c_1$, $c_2$, which denote the cognitive and social parameters respectively are set to 2, 1 and $r_1$, $r_2$ are random real numbers which are drawn from the uniformly distributed interval $[0,1]$. 

III. VORONOI NEIGHBORHOOD IN PSO

A. Voronoi neighborhood motivations

As introduced in section 1, the particle swarm optimization algorithm is based on the swarm intelligence of bird flocking, fish schooling, and other similar social behavior. Reynolds has stated that each bird in a flock flies while steering toward the perceived flock center, avoiding collision, and matching its velocity with its neighbors.

Suppose that dots in Fig.3-a are birds in a flock. According to Reynolds’ thesis all birds must coordinate themselves with their neighbors. Now which ones are the neighbors of the bird in the middle?

One strategy is to draw a circle around it and consider the birds in it as neighbors (Fig.3-b). Now the problem is the radius of the circle. The next problem is that some useful information might be missed. In this case (Fig.3-b) some information from the birds below is missed while in a real bird flock all the information is retrieved and taken into account.

On the one hand the radius must not be so small that we have loss of information, and on the other hand must not be so large that we have useless information. For example, in Fig.3 the bird in the top has no new information for the bird in the middle. Its information can be retrieved from other neighbors in between.

Our motivation to use Voronoi diagram in order to define a new neighborhood in PSO was its useful property that there exists an empty passing circle through two points if they are adjacent in Voronoi diagram, no matter what the distance – Fig.3-c.

B. PSO with Voronoi Neighborhood (VPSO)

As stated in section 2.1 and Definition 2, the Voronoi neighborhood can be determined easily by the Voronoi diagram. According to the fact that the most rational and natural neighborhood is Voronoi neighborhood; one can rewrite the PSO algorithm as follows:

1. Initialize particle positions in n-dimensional space and their velocities with some random real number generator.
2. Calculate the Voronoi neighborhood of each particle.
3. Select the best neighbor of each particle.
4. Update each particle position and velocity according to equations 4, 5.
5. For each particle $i$ if the new position is better than $P_i$ (best position encountered by particle $i$ so far), update $P_i$ to the new position.
6. Repeat steps 2 to 6 until the termination criteria are met.

For other types of neighborhood, only step 2 must change.

C. PSO with Dynamic Euclidean Neighborhood

As we touched on in section 3.1, the static Euclidean neighborhood has some problems about the radius of the circle for determining the neighbors. We can solve this problem by considering the dynamic radius for each iteration.

In this approach, at each iteration, a fixed radius is considered for all particles, and the particles that lie within the circle surrounding each particle are considered as its neighbors. This radius is dynamically calculated over different iterations and is based on the maximum distance between the particles (equation 6) where “dim” means dimension of search space and $N$ is the number of particles.

$$r = \max(d(X_i, X_j)) \frac{d}{\sqrt{N}}, \quad i, j = 1, 2, ..., N \quad i \neq j$$

(6)
IV. EMPIRICAL RESULT

We have run the PSO algorithm with some different configurations such as different neighborhood types and swarm sizes, on some famous optimization test functions in n-dimensional space (Table I) to satisfy constrains mentioned above. All functions except “GoldsteinPrice” are well-defined in an arbitrary dimension. We chose 2 and 10-dimensional spaces to run PSO on test functions.

To implement the PSO, we used the MATLAB® toolbox named psotb-beta-0.3 available at [22], debugged it, and did some revisions to it. The toolbox itself contained the gbest and lbset (ring) topologies. We added the ability of calculating the Voronoi and dynamic Euclidean neighborhoods to it. We have uploaded this modified code at [23].

The range of initialization of particles is important and sometimes in practical problems we don’t know the range that the optimum is in. So we run the PSO in both cases, the case that the optimum is in the initial range and the case that it is out of the initial range (Table I). Table II shows the result of the first case (i.e. in range initialization) in 2-dimensional space. All methods were run with 10 and 20 particles and the goal was set on 1e-10 (while the optimum value for all test functions is zero). Also the limit of number of iterations was set on 250 for 2-dimensional space.

In the second case (i.e. when the optimum is out of the initial range) in 2-dimensional space, we see that PSO with geometric neighborhood (Voronoi and Euclidean) gains better accuracy than the others (Table III). These results show that geometric neighborhood gives more useful information to explore the environment. The results have been obtained from the same configuration as the previous case.

We also run and compared the performance of all types of PSO in 10-dimensional space. All types were run with 15, 20 and 25 particles and the goal was set on 1e-10 (while the optimum value for all test functions is zero). Also the limit of number of iterations was set on 1000 for 10-dimensional space. Table IV shows the result of the first case (i.e. when the optimum is in the initial range) in

**TABLE I.**

**N-DIMENSIONAL OPTIMIZATION TEST FUNCTIONS**

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
<th>Minimum at</th>
<th>In Range Initial</th>
<th>Out Range Initial</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeJong</td>
<td>[ \sum_{i=1}^{n} x_i^2 ]</td>
<td>((0, 0, \ldots, 0) \in \mathbb{R}^n)</td>
<td>-100 (\leq x_i \leq 100)</td>
<td>100 (\leq x_i \leq 200)</td>
</tr>
<tr>
<td>Griewank</td>
<td>[ \sum_{i=1}^{n} \frac{x_i^2}{4000} + \frac{1}{\sum_{i=1}^{n} \cos \left( \frac{x_i}{\sqrt{n}} \right)} + 1 ]</td>
<td>((0, 0, \ldots, 0) \in \mathbb{R}^n)</td>
<td>-100 (\leq x_i \leq 100)</td>
<td>100 (\leq x_i \leq 200)</td>
</tr>
<tr>
<td>Rastrigrin</td>
<td>[ 100n + \sum_{i=1}^{n} (x_i^2 - 10 \cos(2\pi x_i)) ]</td>
<td>((0, 0, \ldots, 0) \in \mathbb{R}^n)</td>
<td>-100 (\leq x_i \leq 100)</td>
<td>100 (\leq x_i \leq 200)</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>[ \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 ]</td>
<td>((1,1,\ldots,1) \in \mathbb{R}^n)</td>
<td>-100 (\leq x_i \leq 100)</td>
<td>100 (\leq x_i \leq 200)</td>
</tr>
<tr>
<td>Easom</td>
<td>[ 1 - \left( \prod_{i=1}^{n} \cos \left( \frac{x_i}{\sqrt{n}} \right) \right) \cdot e^{-\left(\frac{\left(\sum_{i=1}^{n} x_i\right)^2}{400}\right)} ]</td>
<td>((0,0,\ldots,0) \in \mathbb{R}^n)</td>
<td>-5 (\leq x_i \leq 5)</td>
<td>1 (\leq x_i \leq 10)</td>
</tr>
<tr>
<td>SODP (Sum of Different Power)</td>
<td>[ \sum_{i=1}^{n}</td>
<td>x_i</td>
<td>^{(i+1)} ]</td>
<td>((0,0,\ldots,0) \in \mathbb{R}^n)</td>
</tr>
</tbody>
</table>

**TABLE II.**

**THE MINIMUM POINT FOUND BY PSO IN 2D SPACE WHEN THE OPTIMUM IS IN THE INITIAL RANGE**

<table>
<thead>
<tr>
<th>Function</th>
<th>Swarm Size</th>
<th>Neighborhood</th>
<th>(x_i)</th>
<th>(x_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GoldsteinPrice</td>
<td>10</td>
<td>gbest</td>
<td>154.04</td>
<td>11.408</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>lbset</td>
<td>48.008</td>
<td>1.2095</td>
</tr>
<tr>
<td>DeJong</td>
<td>10</td>
<td>gbest</td>
<td>0.1463</td>
<td>0.0004</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>lbset</td>
<td>0.0234</td>
<td>0.0013</td>
</tr>
<tr>
<td>Rastrigrin</td>
<td>10</td>
<td>gbest</td>
<td>2.8977</td>
<td>1.578</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>lbset</td>
<td>1.8523</td>
<td>0.7558</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>10</td>
<td>gbest</td>
<td>4.7890</td>
<td>1.2099</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>lbset</td>
<td>2.4391</td>
<td>0.7309</td>
</tr>
<tr>
<td>Griewank</td>
<td>10</td>
<td>gbest</td>
<td>0.0247</td>
<td>0.0338</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>lbset</td>
<td>0.0162</td>
<td>0.0226</td>
</tr>
<tr>
<td>Easom</td>
<td>10</td>
<td>gbest</td>
<td>0.0422</td>
<td>0.3130</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>lbset</td>
<td>0.0238</td>
<td>0.1521</td>
</tr>
<tr>
<td>SODP</td>
<td>10</td>
<td>gbest</td>
<td>0.0210</td>
<td>9.9985</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>lbset</td>
<td>0.1904</td>
<td>2.53e-6</td>
</tr>
</tbody>
</table>
10-dimensional space. Also Table V contains the result of the second case (i.e. when the optimum is out of the initial range).

### TABLE III.
The minimum point found by PSO in 2D space when the optimum is out of the initial range

<table>
<thead>
<tr>
<th>Function</th>
<th>Swarm Size</th>
<th>Neighborhood</th>
<th>gbest</th>
<th>lbest</th>
<th>Voronoi</th>
<th>Euclidean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goldstein Price</td>
<td>15</td>
<td>reached</td>
<td>1.68e-8</td>
<td>reached</td>
<td>reached</td>
<td>reached</td>
</tr>
<tr>
<td>DeJong</td>
<td>20</td>
<td>reached</td>
<td>9.77e-9</td>
<td>reached</td>
<td>reached</td>
<td>reached</td>
</tr>
<tr>
<td>Rastrigrin</td>
<td>15</td>
<td>9.49e-6</td>
<td>27.108</td>
<td>50.802</td>
<td>29.876</td>
<td></td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>15</td>
<td>10.945</td>
<td>20.914</td>
<td>9.4824</td>
<td>31.641</td>
<td></td>
</tr>
<tr>
<td>Griewank</td>
<td>25</td>
<td>9.58e-6</td>
<td>1.97e-1</td>
<td>0.0437</td>
<td>0.7168</td>
<td></td>
</tr>
<tr>
<td>Easom</td>
<td>25</td>
<td>0.0467</td>
<td>0.0172</td>
<td>0.4522</td>
<td>0.4528</td>
<td></td>
</tr>
<tr>
<td>SODP</td>
<td>15</td>
<td>3.49e-6</td>
<td>1</td>
<td>0.0259</td>
<td>0.9468</td>
<td></td>
</tr>
</tbody>
</table>

† “reached” means that PSO reached the goal (i.e. 1e-10)

Among all kinds of neighborhood (i.e. lbest, Voronoi and Euclidean), Voronoi has the best performance overall but on Easom function gbest (which only consider the global best) gained a better result. Because this function has only one minimum and gbest has the fastest convergence speed. In fact, each kind of neighborhood decreases the risk of getting stuck in local minimum but slows down the speed of convergence. So because there is no local minimum in Easom function and gbest is the fastest, it gains better result in 1000 iterations. In contrast, on some functions which have local minimum the result of gbest is not as good as the previous case. For instance, in Table V, you can see that how gbest got stuck in local minimum on Rosenbrock function.

### V. CONCLUSION

This paper introduced new geometric methods to choose neighbors in PSO algorithm that could yield higher accuracy. These kinds of neighborhood are more reasonable according to realistic birds flocking.

Different types of neighborhood affect the scattering of neighbors of each particle. The lbest neighborhood highly scatters the particles over the environment and leads to gathering unnecessary information to find the optimum point. In contrast to lbest, Euclidean neighborhood with bounded circle area to choose neighbors result in losing useful information. But Voronoi neighborhood gives a proper strategy to avoid gathering extra information or losing useful information. This property lead to find better optimum points, especially when the desired optimum point is out of initialization range.

The only constraint against VPSO is the running time of the algorithm which depends on the dimension of search space. By increasing the dimension the running time of the computing the Voronoi neighbors will increase. As a future work, we are going to use approximate methods to determine Voronoi neighbors of each particle at each iteration, for high-dimensional spaces.

### REFERENCES


