International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)
Volume 4, Issue 2, 2017, PP 29-37
ISSN 2349-4840 (Print) & ISSN 2349-4859 (Online)
DOI: http://dx.doi.org/10.20431/2349-4859.0402005
www.arcjournals.org

PSO-Based Optimization of 2D Streamline Placement

Mebarki A., Houache A., Moufok-Saadoun L., Bouguenous Y., Saad El Hachemi A.

Université des Sciences et de la Technologie d'Oran – Mohamed Boudiaf, Oran, Algeria

Abstract: *In scientific visualization, we often handle vector or direction fields. An efficient way of visualizing such a vector or direction field is to cover the whole domain studied by a set of streamlines because they give good information about the flow and the topology of the underlying field.*

In this article, we propose to optimize already constructed streamlines by defining an optimization criterion related to the length of the line. We propose to use the Particle Swarm Optimization (PSO) method. It makes it possible by inducing the results to converge to the optimum of the objective function related to the defined problem.

The obtained results are encouraging and merit continuity in the enrichment of the objective function.

Keywords: *Streamline, Particle Swarm, Optimization.*

1. INTRODUCTION

Vector or direction fields are commonly used in scientific visualization. They associate a vector or a direction at each point of the domain space and characterize, for example, the velocity and direction of a fluid or the force and the direction of a magnetic or gravitational force.

An efficient way of visualize such a vector or direction field is to cover the whole space domain by a set of streamlines because they give a good information about the flow and the topology of the underlying field.

A streamline has, at each point a tangent equivalent to the vector or parallel to the direction of the corresponding field. In the case of a velocity field of a fluid, it corresponds to the trajectory of a massless particle driven by the flow. If the flow is stationary, the particle continuously follows the same trajectory, thus generating the same streamline.

Bilinear interpolation is an interpolation method on a regular grid. It makes possible to calculate the value of a function from its four nearest neighbors at any point in the domain.

It is a method widely used in digital imaging for image resizing, which allows obtaining better results than the interpolation by the nearest neighbor, while maintaining reasonable complexity.

To get the vector field value at a given point p, we have just to interpolate the value p_v from the already known values $p_1\ p_2\ p_3\ p_4$ which are the closest points of the domain where the value of the field must be known (See Figure).

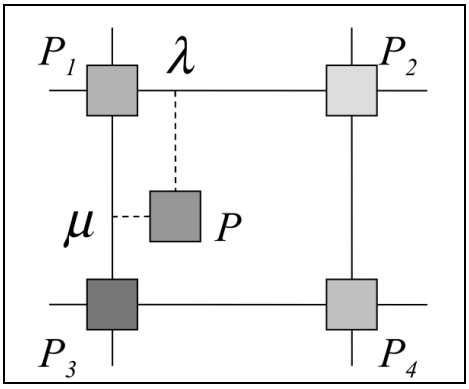

Figure1. *Interpolation Scheme Definition*

The value of the new point is given by the following equation:

$$(1-\mu)\,p1 + \mu p4] + \lambda\,[(1-\mu)\,p2 + \mu p4]$$

Where λ and μ are two real numbers between *0* and *1*defining the position of *p* according to the two domain axes (See Figure).

For a stationary field, a streamline is similar to the trajectory of a massless particle carried by the flow. Given a function *v(p)*which gives the value of the vector at any point *p* of the domain (by interpolation if necessary), this trajectory is obtained by solving the following differential equation:

$$\frac{dP}{dt} = V(p(t)), p(0) = p_0 \qquad \textit{Equation (1)}$$

Where: *p(t)*is the position of the point *p* at the instant *t*; And p_0 is the starting point of the trajectory.

The position of the particle after a time interval Δt is determined by the integral:

$$p(t + \Delta t) = p(t) + \int_t^{t+\Delta t} V(p(t))dt \qquad \textit{Equation (2)}$$

Several numerical integration methods have been proposed to solve Equation(2).

By solving this integral iteratively, we obtain a sequence of points $\{p_k, 0 < k < n\}$, which represents the trajectory of the particle.

The Runge-Kutta method is a second order numerical method that approximates the Equation(2) resolution.

This method allows us to improve the precision of the Euler integrator by introducing an intermediate point p'_k which is calculated by the following equation (midpoint equation):

$$p_k^{'} = p_k + \frac{1}{2} * h * v(p_k)$$

Then we compute the following point P_{k+1}:

$$p_{k+1} = p_k + h * v(p'_k)$$

These definitions could be applied directly to generate greedy placement of streamlines. In this paper, we present an optimization approach to improve the placement quality according to streamline lengths. Longer streamlines are always preferred instead of shorter ones because they give a good view of the topology of the underlying flow.

The rest of this paper is organized as follows: First, we present a short bibliography of streamline placement algorithms. Then, we present the original PSO method. Finally, we describe our idea to optimize streamline placement using PSO approach.

2. PREVIOUS WORK

Streamlines are ones of the first tools used to visualize the directional information of a vector field. Due to their very frequent use, several works have focused on improving the precision and the speed of their calculations.

In 1996, *Turk and Banks* [1] proposed an iterative streamline algorithm. The domain is first covered by a randomly placed streamlines then the image is improved by successive refinements. The allowed changes on this set of streamlines are move, elongate, connect, create, and destroy operations. The interest of this method is the harmonious placement of relatively long streams, imitating the style of freehand illustrations. Its major disadvantage is the slow process of optimization which modifies each streamline randomly and only validates the change if the overall energy of the image is diminished. It should also be noted that the algorithm does not end up with the optimization at a deemed sufficient stage (no criterion of global evaluation of the quality of the obtained image).

Jobar and Lefer developed an alternative algorithm to produce the same results in Turk and Banks in a much more efficient way [2].Their algorithm consists of a long and uniformly spaced streamlines (a minimum separation distance defined by the user between the control points to define the density of the field). They must create a first line and insert it in a queue. Then for each control point of each streamline in the queue, a search is performed for a valid seed point. After selecting a seed point that is quite distant from the already built streamlines (respecting the density), they integrate a streamline

until the latter comes out of the domain, or exceed the density criterion. The process is repeated until no more seed points can be found.

To give more importance to the topology of the flow, *Verma and al.* start the integration of the streamlines in the vicinity of the critical points. This approach produces visualizations that best reflect the topology of the field but does not impose any constraints on the density or on the length of the lines. Moreover, the authors choose a random method to choose the seeds after having saturated the vicinity of the critical points [3].

Farthest Point Seeding Strategy tries to improves the quality of the streamlines, while, at the same time, obtain significant gains in efficiency, robustness and simplification of parameter numbers. Streamlines are approximated by polylines, the points of which are inserted in a 2D Delaunay triangulation. The empty circles defined by Delaunay's triangles provide us with a good approximation of the cavities in the domain [4].

First, it is necessary to insert in the Delaunay triangulation a set of points sampled on the edge of a square bounding the visualization domain. The aim of this step is to delimit the visualization domain. Without this initialization, we would have no information about the regions between the streamlines and the closest of the borders. To locate and measure the distances between the points, they used the diameters of the circumscribed circles to the triangles of the Delaunay triangulation built from the set of integrated points.

Mebarki proposed a modified version of the Farthest Point Strategy using a simple data structure instead of the Delaunay triangulation. He presents an adaptive distance grid to model the visualization domain and get anywhere the local distance to all the other streamlines and the boundaries exactly without any approximation [15].

Many works exist to illustrate 3D placements of streamlines; almost of them are simple extensions of previously cited 2D works. The reader can refer to this paper references to get more information [6,7,8,9,10,11].

3. SWARM PARTICLE OPTIMIZATION

Swarm intelligence is a family of metaheuristics inspired by natural phenomena and more specifically by the behavior of a group or a population of agents that communicate between themselves and interact with their environment in order to survive. These interactions allow the population of agents to perform complex tasks in an organized way [12].

In this paper, we are interested by Particle Swarm Optimization (PSO), a population-based method that was first introduced by *Russel Eberhart* and *James Kennedy* in 1995 [13]. The model was later extended to a simple and efficient optimization algorithm in several domains [14].

Particle Swarm Optimization draws heavily on the gregarious relationships of migratory birds that have long distances to fly, and have to optimize their movements in terms of energy expended, such as *V* formation.

a. The Principle of the Method

The principle of the algorithm is to move the particles to the optimum. Each of these particles, which is a possible solution, is provided with:

- A position, that is, its coordinates throughout the domain definition.
- A neighborhood, that is, a set of particles that interact directly with the particle, especially the one with the best criterion.
- A velocity that allows the particle to move. In this way, during the iterations, each particle changes its position. It evolves according to its best neighbor, its best position, and its previous position. It is this evolution that makes it possible to fall on an optimum particle (See Figure).

Changing the velocity of the particle based on its data and that of the neighbors implies:

- An adventurous trend to continue at current velocity.
- Or a conservative trend, bringing more or less the best position already found.

At every moment, each particle knows:

- Its best position visited. We retain essentially the value of the calculated criterion as well as its coordinates.
- The position of the best neighbor of the swarm that corresponds to the optimal scheduling.
- The value it gives to the objective function because, at each iteration, it is necessary to compare the value of the criterion given by the current particle with the optimal value.

Three types of behavior influence the displacement of a particle:

- A physical component: the particle tends to follow its own path.
- A cognitive component: the particle tends to return to the best site by which it has already passed.
- A social component: the particle tends to move towards the best site already reached by its neighbors.

The solutions found by the particles at a neighborhood represent local optimums, from these local optimums, and by propagating the best solution if it has been reached towards the other neighborhoods, the set of particles will normally, converge towards the optimal global solution of the corresponding problem.

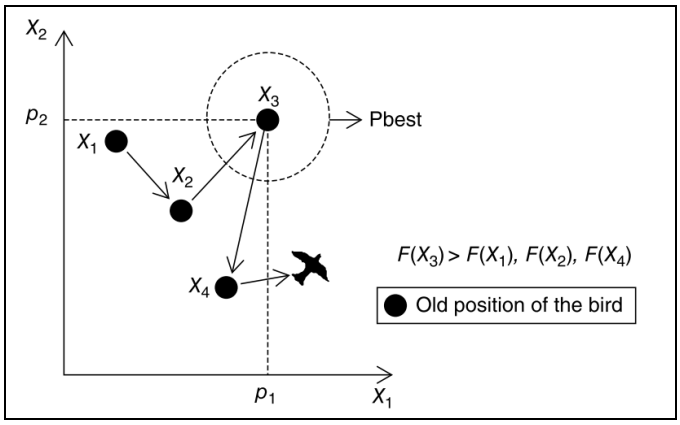

Figure2. *Concepts of the best individual position in a two-dimensional maximization problem [15]*

b. Formal Definition of the Problem

The basic algorithm of the PSO works on a population called a swarm of possible solutions, themselves called particles. These particles (*M* particles) are placed randomly in the search space of the objective function.

At each iteration, the particles move, taking into account their best position (selfish displacement) but also the best position of its neighborhood (Panurgian displacement). In fact, the new velocityis calculated from the following formula:

$$v_{k+1} = w * v_k + c_1 * (Pos_{best} - Pos_{curr}) + c_2 * (NPos_{be} \quad \textit{Equation (3)}$$

- v_{k+1} and v_k are the velocities of the particle at the iterations *k* and *k+1*.
- Pos_{best} is the best position of the particle.
- $NPos_{best}$ is the best position of its neighborhood at iteration *k*.
- Pos_{curr} is the position of the particle at the iteration *k*.
- *w* is the coefficient of inertia
- c_1 and c_2 are the acceleration coefficients

The next position of the particle can then be determined using the velocity just calculated:

$$x_{k+1} = x_k + v_{k+1}$$

Where x_k and x_{k+1} are the positions of the particle at the iteration *k* and *k+1*.

c. The Algorithm

The following pseudo code describes the basic version of the PSO algorithm:

1. Generate M initial solutions and evaluate their fitness values (objective function values)
2. Initialize the velocities of all solutions randomly
3. **For** j=1 **to** M
   Put $PBest_j = solution_j$
4. **Set** GBest = the best solution in the population
5. **While** (termination criteria not satisfied)
   i. **For** j=1 **to**M
   If (Fitness ($Solution_j$) is better than PBestj)
   $PBest_j = solution_j$
   If ($PBest_j$ is better than GBest)
   GBest = $PBest_j$
   ii. **For** j=1 **to** M
   1. Velocity($solution_j$)
   2. Control Velocity($solution_j$)
   3. Update $Solution_j$
   iii. Evaluate Fitness for all solutions
   iv. Update inertia weight parameter w
6. Report the best solution

4. NEIGHBORHOOD DEFINITION

The neighborhood describes the structure of the population. The particles within a neighborhood communicate with one another. Several neighborhoods have been studied and are considered in terms of particle identifiers and not only topological information such as Euclidean distances in the search space [16,17] (See Figure).

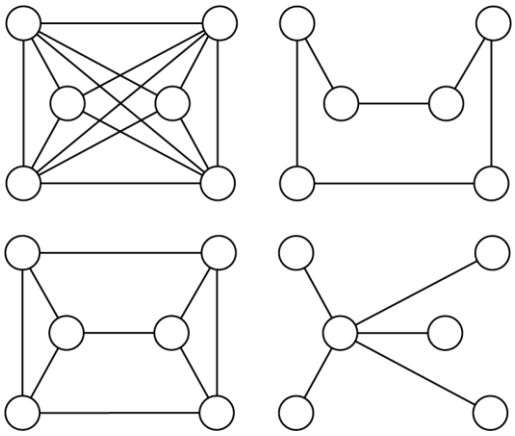

Figure3. *Four kinds of neighborhood topologies: (1) Fully connected Topology, (2) Ring Topology, (3) Mesh Topology, (4) Star Topology*

a. Ring Topology

In this kind of topology, neighbors are tightly connected so they react when one fitness particle raises, this reaction weakens proportionally with respect to the distance. Thus, one subdivision of the population might converge to a local optimum, while another subdivision might converge to a different point or remain searching. However, the optima will eventually pull the swarm.

b. Fully Connected Topology

The fully connected topology or the full topology is defined when all nodes are directly connected among each other. This topology is also known as the PSO's GBest version, in which all individuals in the whole swarm direct their flight toward the best particle found in the whole population.

c. Star Topology Mesh Topology

In star topology, the information passes through only one particle. One central node influences and it is influenced by all other members of the swarm.

d. Mesh Topology

In this topology, one node is connected to limited number of other nodes according to their locations. An example of such a topology is to connect each particle to its four direct neighbors. We have to consider redundancy in the research process because of the overlapping neighbors in each particle. In this paper, we adopted this topology in such a way that each streamline is correlated with its direct neighbors.

5. PSO-BASED OPTIMIZATION OF STREAMLINE PLACEMENT

We present now our algorithm for optimizing streamline placements.

The algorithm is based on the PSO approach where the particles are the seed points of the streamlines. Each streamline is integrated from its seed point by successive integrations until:

- It reaches the boundaries of the domain;
- It becomes very close to another streamline.

The initial solution corresponds to the initial placement of the streamlines that are built from a randomly generated set of particles.

The basic algorithm has to be adapted to our context in such a way to allow re-integrating streamlines at each iteration. In each step, and for each particle, we have to remove the corresponding streamline and its neighbors in order to evaluate the new position of the particle and reintegrate it since the neighboring streamlines stop the integration process. Removing neighbors is necessary to permit a maximum lengthening of the streamlines.

The stop criteria is defined as follow:

- The maximum specified number of iterations is reached.
- The value of the objective function is acceptable.
- The velocity variation is close to 0.

a. The Proposed Algorithm

1. Generate N initial solutions and evaluate their fitness values (objective function values)
2. Initialize the velocities of all solutions randomly
3. **For** j=1 **to** M
   Put $PBest_j = solution_j$
4. **Set** GBest = the best solution in the population
5. **While** (termination criteria not satisfied)
   i. **For** j=1 **to**M
      If (Fitness($Solution_j$) is better than PBestj)
         $PBest_j = solution_j$
      If ($PBest_j$ is better than GBest)
         GBest = $PBest_j$
   ii. **For** j=1 **to** M
      i. Remove the current streamline (j) and its neighbors
      ii. Calculate Velocity($solution_j$)
      iii. Control Velocity($solution_j$)
      iv. Reintegrate the current streamline (j) and its neighbors
      v. Update $Solution_j$
   iii. Evaluate Fitness for all solutions
   iv. Update inertia weight parameter w
6. Report the best solution

b. Fitness

In our approach, we considered a simple function that favor long streamlines instead of short ones.

This function is equal to the ratio between streamline for a given particle and the average length of all other streamlines:

$$\frac{\textit{The current stream line length}}{\textit{The average length of all the stream lines}}$$

6. Experimental Results

To experiment our approach, we defined the following configuration scheme for our tests:

- A domain size equal to 512x512 to place streamlines.
- Number of particles: 128 seed points.

We experimented our algorithm with four different synthetic vector fields; the calculations converged after ten iterations. The results are presented inTable and Figure.

Table1. *The convergence of the global objective function for 4 different synthetic vectof fields over 10 iterations*

Iteration	Vector Field 1	Vector Field 2	Vector Field 3	Vector Field 4
1	143,931584	144,630555	143,931584	144,129016
2	149,621779	153,360759	149,621779	145,940026
3	168,965339	159,057542	168,965339	147,447136
4	174,357213	165,593771	174,357213	149,703461
5	179,993769	168,718661	179,993769	149,896458
6	179,99523	171,579886	179,99523	150,261208
7	183,546551	182,058246	183,546551	150,266143
8	185,335604	183,841781	185,335604	150,276948
9	187,871382	183,841781	187,871382	150,276948
10	187,871382	183,841781	188,825062	150,276948

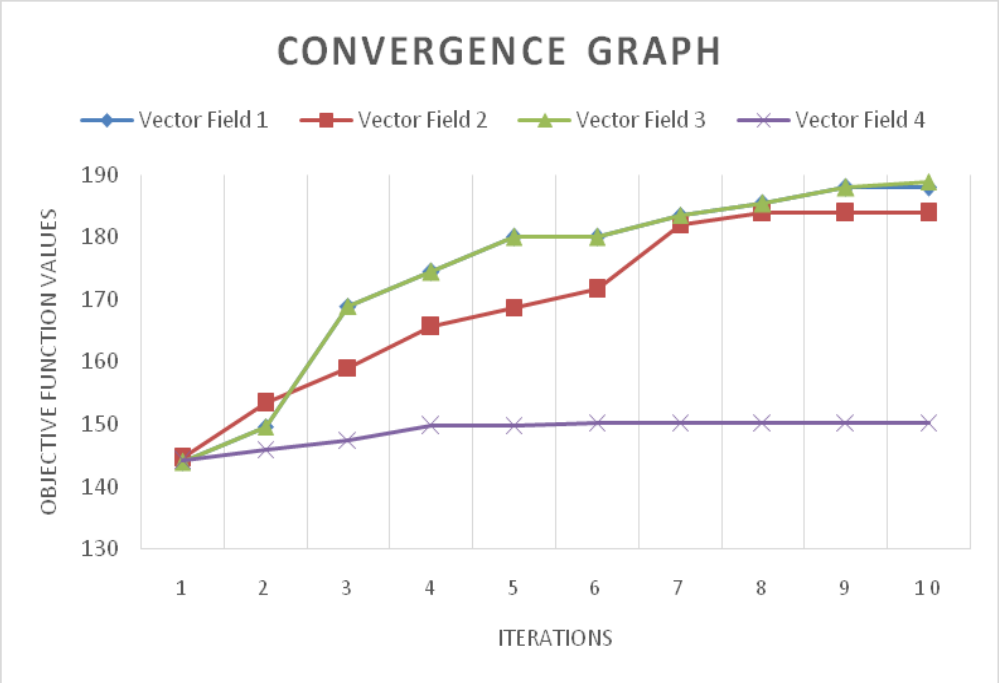

Figure4. *Convergence graph of the test vector fields over 10 iterations*

7. Conclusion

In this paper, we presented an idea to optimize streamline placements. We tried to fit the PSO algorithm to improve the quality of already placed streamlines. The quality is often correlated with streamline lengths. So, we defined our objective function in term of streamline length. The algorithm converge rapidly, and the obtained results are promising and uncourageous.

REFERENCES

[1] G. Turk and D. Banks, "Image-Guided Streamline Placement," ACM SIGGRAPH Conference Proceedings, pp. 453-460, 1996.

[2] Bruno Jobard and Wilfrid Lefer, "Creating Evenly-Spaced Streamlines of Arbitrary Density," in Visualization in Scientific Computing ’97: Springer Vienna, 1997, pp. pp 43-55.

[3] V. Verma, Kao D., and A. Pang, "A flow-guided streamline seeding strategy," in Visualization'00, Salt Lake City, UT, USA, 2000, pp. 163 - 170.

[4] Pierre Alliez, Olivier Devillers, and Abdelkrim Mebarki, "Farthest Point Seeding for Efficient Placement of Streamlines," in Proc. of IEEE Visualization, 2005.

[5] Abdelkrim Mebarki, "Adaptive Distance Grid Based Algorithm for Farthest Point Seeding Streamline Placement," Open Computer Science, vol. 6, no. 1, pp. 91-99, July 2016.

[6] Fuhrmann and E. Groller, "Real-time techniques for 3D flow visualization," in Visualization '98. Proceedings, Research Triangle Park, NC, USA, 24-24 Oct. 1998, pp. 305 - 312. [Online]. https://www.cg.tuwien.ac.at/research/vis/VR/viskas/viskas.pdf

[7] Xiangong Ye, D. Kao, and A. Pang, "Strategy for seeding 3D streamlines," in Visualization, Vis'05, 2005, pp. 471 - 478.

[8] Liya Li and Han-Wei Shen, "Image-Based Streamline Generation and Rendering," Transactions on Visualization and Coputer Graphics, vol. 13, no. 3, May/June 2007.

[9] Benjamin Spencer, "Evenly Spaced Streamlines for Surfaces: An Image-Based Approach," Computer Graphics Forum, vol. 28, no. 6, pp. 1618–1631, September 2009.

[10] Michael Schlemmer, Ingrid Hotz, Bernd Hamann, Florian Morr, and Hans Hagen, "Priority streamlines: a context-based visualization of flow fields," in 9th Joint Eurographics / IEEE VGTC conference on Visualization, 2007, pp. 227-234.

[11] Lijie Xu, Teng-Yok Lee, and Han-Wei Shen, "An Information-Theoretic Framework for Flow Visualization," Visualization and Computer Graphics, IEEE Transactions on, vol. 16, no. 6, pp. 1216 - 1224, Nov.-Dec. 2010.

[12] Russel Eberhart and James Kennedy, "Particle swarm optimization," in IEEE International Conference on Neural Networks, Perth, WA, Australia, 1995, 27 November - 1 December.

[13] Lazinica Aleksandar, Ed., Particle Swarm Optimization. Rijeka, Croatia: In-Tech, 2009.

[14] Omid Bozorg-Haddad, Mohammad Solgi, and Hugo A. Loáiciga, "Meta-Heuristic and Evolutionary Algorithms for Engineering Optimization," , Hoboken, NJ 07030, USA, 2017.

[15] Angelina Jane Reyes Medina, José Gabriel Ramírez-Torres, and Gregorio Toscano Pulido, "A Comparative Study of Neighborhood Topologies for Particle Swarm Optimizers.," in Proceedings of the International Joint Conference on Computational Intelligence, Madeira, Portugal, 2009, October 5-7, 2009.

[16] Angel Eduardo Muñoz Zavala, "A Comparison Study of PSO Neighborhoods," in A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation II. Advances in Intelligent Systems and Computing, Oliver Schütze et al., Eds. Berlin, Heidelberg: Springer, 2013, pp. 251-265.

[17] Olufemi Rosanwo, Christoph Petz, Steffen Prohaska, Hans-Christian Hege, and Ingrid Hotz, "Dual Streamline Seeding," in Visualization Symposium, 2009, pp. 9-16.

[18] Maurice Clerc and James Kennedy, "The Particle Swarm—Explosion, Stability, and Convergence in a Multidimensional Complex Space," IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, vol. 6, no. 1, pp. 58-73, 2002.

[19] "Particle Swarm Optimization with Inertia Weight and Constriction Factor," in ICSI: International conference on swarm intelligence, Cergy, France, 2011, 14-15 june.

AUTHOR'S BIOGRAPHY

Abdelkrim Mebarki, is an assistant professor at Informatics Department in Oran University of Science and Technology. He has his Ph.D. in Computer Science from Nice-Sophia Antipolis University. His research interests are about Geometric Data Structures, Scientific Visualization and Computer Graphics. He is the author of many books and articles in theses domains.