**Tutorial on particle swarm optimization and its combinations to other evolutionary algorithms**

**Abstract**

**1. Introduction to particle swarm optimization (PSO)**

Particle swarm optimization (PSO) algorithm was developed by James Kennedy (a social psychologist) and Russell C. Eberhart (an electrical engineer). This tutorial is navigated by the article “Particle swarm optimization: An overview” of Riccardo Poli, James Kennedy, and Tim Blackwell. The main idea of PSO is based on social intelligence when it simulates how a flock of birds search for food. Given a target function known as *cost function* *f*(***x***), the optimization problem is to find out the minimum point ***x***\* known as minimizer or optimizer so that *f*(***x***\*) is minimal. In PSO theory, *f*(***x***) is also called *fitness function* and thus, when *f*(***x***) is evaluated at *f*(***x***0) then, *f*(***x***0) is called fitness value which represents the best food source for which a flock of birds search.. If ***x***\* is an optimizer, *f*(***x***\*) is called optimal value, best value, or best fitness value. As a convention, the optimization problem is global minimization problem when ***x***\* is searched over entire domain of *f*(***x***). For global maximization, it is simple to change a little bit our viewpoint.

Traditional local optimization methods such as Newton-Raphson and gradient descent along with global optimization methods require that *f*(***x***) is differentiable. Alternately, PSO does not require existence of differential. PSO scatters a population of candidate solutions (candidate optimizers) for ***x***\* and such population is called swarm whereas each candidate optimizer is called particle in the swarm. PSO is an iterative algorithm running over many iterations in which every particle is moved at each iteration so that it approaches the global optimizer ***x***\*. Movement of all particles is attracted by ***x***\*. In other words, such movement is attracted by minimizing *f*(***x***) so that *f*(***x***) is small enough. In PSO, ***x*** is considered as position of particle. The movement of each particle is affected by its best position and the best position of the swarm. Note, the closer to ***x***\*, the better the position is.

As a formal definition, let be the swam of particles and let ***x****i* and ***p****i* be current position and best position of particle *i*. Note, ***p****i* is called *local best position*. Moreover, the movement speed of particle *i* is specified by its velocity ***v****i*. Let ***p****g* be the *global best position* of entire swarm. The closer to ***x***\*, the better the positions ***p****i* and ***p****g* are. It is expected that ***p****g* is equals to ***x***\* or is approximated to ***x***\*. The ultimate purpose of PSO is to determine ***p****g*.

Of course, ***x****i*, ***p****i*, and ***p****g* are *n*-dimensional points and ***v****i* is *n*-dimensional vector because *f*(***x***) is from real *n*-dimensional space ***R****n* to real space ***R***. Following is pseudo-code of PSO (Wikipedia, 2017).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input: the swam of particles along with their initialized positions and velocities.  Output: the global best position ***p****g* of entire swarm with expectation that ***p****g* is equal or approximated to the global minimizer ***x***\*.  Let ***lb*** and ***ub*** be lower bound and upper bound of particles in their search space. They are vectors.  All current positions ***x****i* of all particles are initialized randomly. Moreover, their best positions are set to be their current positions such that ***p****i* = ***x****i*. Note, all particles are randomized in the range [***lb***, ***ub***] as closed sphere.  All current velocities ***vi*** of all particles are initialized randomly. Because each ***vi*** is vector, its elements are randomized in the range [–|***ub*** – ***lb***|, |***ub*** – ***lb***|] from –|***ub*** – ***lb***| to |***ub*** – ***lb***| where the notation |.| denotes distance between two vectors or two points.  The global best position ***p****g* is assigned by the local best position ***p****i* such that *f*(***p****i*) is smallest among particles.  While terminated condition is not met do  For each particle *i* in swarm *S*  Velocity of particle *i* is updated as follows:   |  |  | | --- | --- | |  | (1.1) |   Position of particle *i* is updated as follows:   |  |  | | --- | --- | |  | (1.2) |   If *f*(***x****i*) < *f*(***p****i*) then  The best position of particle *i* is updated: ***p****i* = ***x****i*  If *f*(***p****i*) < *f*(***p****g*) then  The best position of swarm is updated: ***p****g* = ***p****i*  End if  End if  End for  End while |

**Table 1.1.** Basic particle swarm optimization (PSO) algorithm

Equation 1.1 is the heart of PSO, which is called *velocity update rule*. Equation 1.2 is called *position update rule*. There are two most popular terminated conditions:

1. The cost function at ***p****g* which is evaluated as *f*(***p****g*) is small enough. For example, *f*(***p****g*) is smaller than a small threshold.
2. Or PSO ran over a large enough number of iterations.

Function *U*(0, *ϕ*1) generates a random vector whose elements are random numbers in the range [0, *ϕ*1]. Similarly, function *U*(0, *ϕ*2) generates a random vector whose elements are random numbers in the range [0, *ϕ*2]. For example,

Note, the super script “*T*” indicates transposition operator of vector and matrix. The operator denotes component-wise multiplication of two points (Poli, Kennedy, & Blackwell, 2007, p. 3). For example, given random vector *U*(0, *ϕ*1) = (*r*11, *r*12,…, *r*1*n*)*T* and position ***x****i* = (*xi*1, *xi*2,…, *xin*)*T*, their component-wise multiplication is:

Two components and are considered as attraction forces that push every particle to move. Sources of force and force are the particle *i* itself and its neighbors. Thus, two most important parameters of PSO are *ϕ*1 and *ϕ*2 which represent the two attraction forces. The popular values of them are *ϕ*1 = *ϕ*2 = 1.4962. Parameter *ϕ*1 along with the force express the *exploitation* of PSO whereas parameter *ϕ*2 along with the force express the *exploration* of PSO (Poli, Kennedy, & Blackwell, 2007, p. 4). The larger parameter *ϕ*1 is, the faster PSO converges but it trends to converge at local minimizer. In opposite, if parameter *ϕ*2 is large, convergence to local minimizer will be avoided in order to achieve better global optimizer but convergence speed is decreased. Parameters *ϕ*1 and *ϕ*2 are also called acceleration coefficients or *attraction coefficients*. Especially, *ϕ*1 is called cognitive weight and *ϕ*2 is called social weight because *ϕ*1 reflects thinking of particle itself in moving and *ϕ*2 reflects influence of entire swarm on every particle in moving. In practical, velocity ***v****i* can be bounded in the range [–***v****max*, +***v****max*] in order to avoid out of convergence trajectories but the parameter ***v****max* is not popular because there are some other parameters such as inertial weight and constriction coefficient (mentioned later) which are used to damp the dynamics of particles. Favorite values for the size of swarm (the number of particles) are ranged from 20 to 50.

Because any movement has inertia, inertial force is added to the two attraction forces. Hence, the inertial force is represented by a so-called *inertial weight* *ω* where 0 < *ω* ≤ 1. Equation 1.1 becomes (Poli, Kennedy, & Blackwell, 2007, p. 4):

|  |  |
| --- | --- |
|  | (1.3) |

The larger inertial weight *ω* is, the faster particles move because its inertial is high, which leads PSO to explore global optimizer. Note that moving fast does not imply fast convergence. In opposite, the smaller *ω* leads PSO to exploit local optimizer. In general, large *ω* expresses exploration and small *ω* expresses exploitation. The inverse 1–*ω* is known as friction coefficient. The popular value of *ω* is 0.7298.

Pioneers in PSO (Poli, Kennedy, & Blackwell, 2007, p. 5) recognized that if velocities ***v****i* of particles are not restricted, their movements can be out of convergence trajectories at unacceptable levels. Therefore, they proposed a so-called *constriction coefficient* *χ* to damp dynamics of particles. Note, *χ* is also called constriction weightor damping weight where 0 < *χ* ≤ 1. With support of constrictioncoefficient, equation 1.1 becomes (Poli, Kennedy, & Blackwell, 2007, p. 5):

|  |  |
| --- | --- |
|  | (1.4) |

It is easy to recognize that equation 1.2 is special case of equation 1.3 when the expression *χ****v****i* is equivalent to the expression *ω****v****i*. The popular value of constriction coefficient is *χ* = 0.7298 given *ϕ*1 = *ϕ*2 = 2.05 and *ω* = 1. Note, inertial weight *ω* is also the parameter that damps dynamics of particles. This is the reason that *ω* = 1 when *χ* ≠ 1 but constriction of *χ* is stronger than *ω* because *χ* affects previous velocity and two attraction forces whereas *ω* affects only previous velocity.

Structure of swarm which is determined by defining neighbors and neighborhood of every particle is called *swarm topology* or *population topology*. Because ***p****g* is the best position of entire swarm, the attraction force indicates the movement of each particle is affected by all other particles, which means that every particle connects to all remaining particles. In other words, neighbors of a particle are all other particles, which is known as fully connected swarm topology. For easily understandable explanation, suppose particles are vertices of a graph, fully connected swarm topology implies that such graph is fully connected, in which all vertices are connected together. Alternately, swarm topology can be defined in different way so that each particle *i* only connects with a limit number *Ki* of other particles. In other words, each particle has only *Ki* neighbors. With custom-defined swarm topology, equation 1.4 is written as follows (Poli, Kennedy, & Blackwell, 2007, p. 6):

|  |  |
| --- | --- |
|  | (1.5) |

Where ***q****k* is the best position of the *k*th neighbor of particle *i*. Of course, ***q****k* is ***p****j* of some particle *j*.

***q****k* = ***p****j* such that particle *j* is the *k*th neighbor of particle *i*.

Please pay attention that, in equation 1.5, particle *i* is also its neighbor. In other words, in equation 1.5, the set of *Ki* neighbors includes particle *i*. The two parameters *ϕ*1 and *ϕ*2 are reduced into only one parameter *ϕ* > 0, which implies the strengths of all attraction forces from all neighbors on particle *i* are equal. The popular value of *ϕ* is *ϕ* = 4.1 given *χ* = 0.7298. Equation 1.5 is known as Mendes’ fully informed particle swarm (FIPS) method. The topology in the basic PSO specified by equation 1.1, equation 1.3, and equation 1.4 is known *global best topology* because only one best position ***p****g* of entire swarm is kept track. However, equation 1.5 indicates that many best positions from groups implied by neighbors are kept track. Hence, FIPS method specifies a so-called *local best topology*, which converges slowly but avoids converging at local optimizer. In other words, local best topology aims to exploration rather than exploitation.

If we focus on the fact that the attraction force issued by the particle *i* itself is equivalent to the attraction force from the global best position ***p****g* and the other attraction forces from its neighbors ***q****k*, equation 1.5 is modified as follows:

|  |  |
| --- | --- |
|  | (1.6) |

In equation 1.6, the set of *Ki* neighbors does not include particle *i* and so, the three parameters *ϕ*1, *ϕ*2, and *ϕ* are co-existent. Inertial weight *ω* is kept intact too. It is easy to recognize that equation 1.6 is the general form of velocity update rule. Equation 1.6 balances local best topology and global best topology with expectation that convergence speed is improved but convergence to local optimizer can be avoided. In other words, equation 1.6 aims to achieve both exploration and exploitation.

In general, the two main aspects of PSO are exploration and exploitation. The exploration aspect aims to avoid premature converging so as to reach global optimizer whereas the exploitation aspect aims to motivate PSO to converge as fast as possible. Besides exploitation property can help PSO to converge more accurately regardless of local optimizer or global optimizer. These two aspects are equally important. Consequently, two problems corresponding to the exploration and exploitation are *premature problem* and *dynamic problem*. Solutions of premature problem are to improve the exploration and solutions of dynamic problem are to improve the exploitation. Inertial weight and constriction coefficient are common solutions for dynamic problem. Currently, solutions of dynamic problem often relate to tuning coefficients which are PSO parameters. Solutions of premature problem relates to increase dynamic ability of particles such as:

* Dynamic topology.
* Change of fitness function.
* Adaptation includes tuning coefficients, adding particles, removing particles, and changing particle properties.
* Diversity control.

*Dynamic topology* is popular in solving premature problem. The topology from equation 1.1, equation 1.3, equation 1.4, equation 1.5, and equation 1.6 is static (Poli, Kennedy, & Blackwell, 2007, p. 6) because it is kept intact over all iterations of PSO. In other words, neighbors and neighborhood in *static topology* are established fixedly. We will later research dynamic topology in which neighbors and neighborhood are changed at each iteration.

**2. Theoretical analysis of PSO**

PSO is the effective method to solve the optimization problem. Its ideology and practice are reasonable and interesting but there are not fully comprehensive mathematical models for PSO yet (Poli, Kennedy, & Blackwell, 2007, p. 14). However, it is necessary to survey these models to understand PSO and its preeminent features.

**3. Variants of PSO**

The PSO shown in table 1.1 is basic PSO. Recently there are many PSO variants. Some of them aim to improve the basic PSO but the others aim to solve raised problems.

Binary PSO (BPSO) developed by James Kennedy is a simple version of PSO where positions ***x****i* and ***p****i* are binary (0 and 1). After velocity update rule (equation 1.1) is executed, the velocity ***v****i* = (*vi*1, *vi*2,…, *vin*)*T* is in turn squashed into range [0, 1] by squash function (logistic function) as follows (Poli, Kennedy, & Blackwell, 2007, p. 9), (Too, Abdullah, & Saad, 2019, p. 3):

|  |  |
| --- | --- |
|  | (3.1) |

Where *vij* is the *j*th element of ***v****i* = (*vi*1, *vi*2,…, *vin*)*T*. Of course, the squashed value *s*(*vij*) is in range [0, 1]. The main point of BPSO is to modified position update rule as follows (Too, Abdullah, & Saad, 2019, p. 3):

|  |  |
| --- | --- |
|  | (3.2) |

Where *xij* is the *j*th element *xij* of ***x****i* = (*xi*1, *xi*2,…, *xin*)*T* and *r* is a random number in range [0, 1]. In general, position update rule in BPSO is specified by equation 3.2 instead 1.2. Therefore, the purpose of BPSO is to simplify the basic PSO.

Bare bones PSO (BBPSO) developed by James Kennedy and Russell C. Eberhart is also a simple version of PSO where velocity update rule (equation 1.1) is eliminated. In other words, positions ***x****i* are updated based on only previous position and previous best position. Given ***x****i* = (*xi*1, *xi*2,…, *xin*)*T*, ***p****i* = (*pi*1, *pi*2,…, *pin*)*T*, and ***p****g* = (*pg*1, *pg*2,…, *pgn*)*T*, BBPSO assumes that the *j*th element *xij* of ***x****i* follows normal distribution with mean (*pij*+*pgj*)/2 and variance (*pij*–*pgj*)2 (Poli, Kennedy, & Blackwell, 2007, p. 13).

|  |  |
| --- | --- |
|  | (3.3) |

Note, the sign “~” denotes distribution and denotes normal distribution. Thus, position update rule in BBPSO is modified as follows (Pan, Hu, Eberhart, & Chen, 2008, p. 3), (al-Rifaie & Blackwell, 2012, p. 51):

|  |  |
| --- | --- |
| Every *xij* is randomized according to normal distribution | (3.4) |

Obviously, position update rule in BBPSO is specified by equation 3.4 instead 1.2 and there is no velocity update rule. Therefore, the purpose of BPSO is to simplify the basic PSO.

As aforementioned, two main problems of PSO are premature problem and dynamic problem. Solutions of premature problem are to improve the exploration so that PSO is not trapped in local optimizer. Exactly, these solutions relate to increase dynamic ability of particles such as dynamic topology, change of fitness function, adaptation (tuning coefficients, adding particles, removing particles, changing particle properties), and diversity control over iterations. Recall that the topology from equation 1.1, equation 1.3, equation 1.4, equation 1.5, and equation 1.6 is static topology because it is kept intact over all iterations of PSO. Here we research dynamic topology in which neighbors and neighborhood are changed at each iteration. Sugnathan (Poli, Kennedy, & Blackwell, 2007, p. 8) proposed to start PSO with small local best topology with a small number of neighbors and such topology is progressively enlarged with a larger number of neighbors after each iteration until getting the full connected topology known as global best topology. The favorite local best topology is lattice ring.

Peram (Poli, Kennedy, & Blackwell, 2007, p. 8) defined the topology dynamically at each iteration by a so-called fitness distance ratio (FDR). Given target particle *i* and another particle *j*, their FDR is the ratio of the difference between *f*(***x****i*) and *f*(***x****j*) to the Euclidean difference between ***x****i* and ***x****j*.

|  |  |
| --- | --- |
|  | (3.5) |

Given target particle *i*, if FDR(***x****i*, ***x****j*) is larger than a threshold (> 1), the particle *j* is a neighbor of the target particle *i*. Alternately, top *K* particles whose FDR (s) with ***x****i* are largest are *K* neighbors of particle *i*.

Clerc (Poli, Kennedy, & Blackwell, 2007, p. 8) proposed a so-called TRIBES algorithm which is a dynamic topology PSO. TRIBES divides the entire swarm into sub-populations called “tribes”. A good tribe has good particles whose best values (best fitness values) *f*(***p****i*) are small enough whereas a bad tribe has bad particles whose best values (best fitness values) *f*(***p****i*) are not small enough. After some iterations, a good tribe can remove particles that are not good enough in order to maintains its preeminence and a bad tribe can add more particles to increase its possibility of improvement. Because TRIBES adds and removes dynamically particles, it can be classified into adaptation solution for dynamic problem.

Recall that premature problem is solved by many solutions such as dynamic topology, change of fitness function, adaptation (tuning coefficients, adding particles, removing particles, changing particle properties), and diversity control over iterations. Here we research the ways which change fitness function evaluation. Noisy adder is an interesting way to change how to evaluate fitness function, but fitness function is indeed not changed. For instance, any time at any iteration when *f*(***x***) is evaluated, a random noise is added into the evaluated result.

The noise *ε* often conforms normal distribution. When ***x****i* is evaluated more than one time, evaluated results can be different. Consequently, it is possible to avoid converging local optimizer while the effectiveness in convergence is kept (Poli, Kennedy, & Blackwell, 2007, p. 10).

As aforementioned, *diversity control* is a solution of premature problem to prevent PSO from converging to local optimizer because a reason of local trapping is that many particles are clustered too tight into one region. Hence, we research here the diversity control.

Recall that two main aspects of PSO are exploration and exploitation. Exploitation is as important as exploration because it asserts success of convergence and speed of convergence. Aforementioned BPSO and BBPSO improve the exploitation. Some algorithms mentioned in next section which are combinations of PSO and other evolutionary algorithms also aim to improve the exploitation.

**3. PSO and other evolutionary algorithms**

Evolutionary algorithms (EA) are the ones which simulate natural activities in biological world. PSO is an EA because it simulates how a flock of birds search for food and so its combination to other EA is natural. Some combinations are so tight that it is possible to form hybrid PSOs which are considered as variants of PSO mentioned in previous section.

**4. Conclusions**

**References**

al-Rifaie, M. M., & Blackwell, T. (2012). Bare Bones Particle Swarms with Jumps. In M. Dorigo, M. Birattari, C. Blum, A. L. Christensen, A. P. Engelbrecht, R. Groß, & T. Stützle (Ed.), *International Conference on Swarm Intelligence.* *Lecture Notes in Computer Science 7461*, pp. 49-60. Brussels: Springer Berlin. doi:10.1007/978-3-642-32650-9\_5

Bonyadi, M. R., & Michalewicz, Z. (2017, March 2). Particle Swarm Optimization for Single Objective Continuous Space Problems: A Review. (N. Lindsay, Ed.) *Evolutionary Computation, 25*(1), 1-54. doi:10.1162/EVCO\_r\_00180

Pan, F., Hu, X., Eberhart, R., & Chen, Y. (2008, September 21). An Analysis of Bare Bones Particle Swarm. *IEEE Swarm Intelligence Symposium 2008 (SIS 2008)* (pp. 1-5). St. Louis, MO, US: IEEE. doi:10.1109/SIS.2008.4668301

Poli, R., Kennedy, J., & Blackwell, T. (2007, June). Particle swarm optimization. (M. Dorigo, Ed.) *Swarm Intelligence, 1*(1), 33-57. doi:10.1007/s11721-007-0002-0

Too, J., Abdullah, A. R., & Saad, N. M. (2019, May 8). A New Co-Evolution Binary Particle Swarm Optimization with Multiple Inertia Weight Strategy for Feature Selection. (A. Bryant, R. Theron, K. Sedig, & D. J. Lizotte, Eds.) *informatics, 6*(2), 1-14. doi:10.3390/informatics6020021

Wikipedia. (2017, March 7). *Particle swarm optimization*. (Wikimedia Foundation) Retrieved April 8, 2017, from Wikipedia website: https://en.wikipedia.org/wiki/Particle\_swarm\_optimization

Zhang, Y., Wang, S., & Ji, G. (2015, October 7). A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications. (G. Xie, Ed.) *Mathematical Problems in Engineering, 2015*(Special Issue: Artificial Intelligence and Its Applications 2014), 1-38. doi:10.1155/2015/931256