**Introduction of particle swarm optimization algorithm and its applications to other evolutionary algorithms**

**Abstract**

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

The main idea of particle swarm optimization (PSO) algorithm 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***), recall that the optimization problem is to find out the minimum point ***x***\* known as minimizer or optimizer so that *f*(***x***\*) is minimal. As a convention, the optimization problem is global minimization problem. For 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. It is focused that 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*. Moreover, the movement speed of particle *i* is specified by its velocity ***v****i*. The cost function *f*(***x***) is from real *n*-dimensional space ***R****n* to real space ***R***. Let ***p****g* be the 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. Following is pseudo-code of PSO (Wikipedia, 2017).

|  |  |  |
| --- | --- | --- |
| Input: the swam of particles along with their initialized positions and velocities.  Output: the best position ***p****g* of entire swarm with expectation that ***p****g* is equal or approximated to the global minimizer ***x***\*.  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*.  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 *S*  Velocity of particle *i* is updated as follows:   |  |  | | --- | --- | |  | (1.1) |   Position of particle *i* is updated as follows:  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 equation*. 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 itself and its neighbors. Thus, two most important parameters of PSO are *ϕ*1 and *ϕ*2 which represent the two attraction forces. They are in the interval (0, 1), which means that 0 < *ϕ*1 < 1 and 0 < *ϕ*2 < 1. 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 are also called acceleration coefficients or *attraction coefficients*.

Because any movement has inertia, inertial force is added to the two at 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):

|  |  |
| --- | --- |
|  | (1.2) |

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* express exploration and small *ω* expresses exploitation. The inverse 1 – *ω* is known as friction coefficient.

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 restriction coefficient *χ* to damping dynamics of particles. With support of restriction coefficient, equation 1.1 becomes (Poli, Kennedy, & Blackwell, 2007, p. 5):

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

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*.

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 particles 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 1.3 is written as follows (Poli, Kennedy, & Blackwell, 2007, p. 6):

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

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*. Please pay attention that, particle *i* is also its neighbor. In other words, the set of *Ki* neighbors include particle *i*. The two parameters *ϕ*1 and *ϕ*2 are reduced into only one parameter *ϕ* where 0 < *ϕ* < 1, which implies the strengths of all attraction forces from all neighbors on particle *i* are equal. It is easy to recognize that equation 1.4 is the most general form of velocity update equation. The swarm topology in equation 1.4 is static (Poli, Kennedy, & Blackwell, 2007, p. 6) because it is kept intact over all iterations of PSO. We will research dynamic swarm topology which is changed at each iteration later.

**2. Variants of PSO**

**3. PSO and other evolutionary algorithms**

**4. Conclusions**

**References**