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

**Abstract**

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

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 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* is 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.

|  |
| --- |
| 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:  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 |

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.

Note, denotes component-wise multiplication of two points. For example, given two vectors ***x****i* = (*xi*1, *xi*2,…, *xin*)*T* and ***x****j* = (*xj*1, *xi*2,…, *xjn*)*T*, their component-wise multiplication is:

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].

**2. Variants of PSO**

**3. PSO and other evolutionary algorithms**

**4. Conclusions**

**References**