Particle Swarm Optimization

Computer Science 760

Patricia J Riddle

What is PSO?

Particle swarm optimization (PSO) is a swarm intelligence based algorithm

- finds a solution to an optimization problem in a search space
- predicts social behavior in the presence of objectives.

PSO is **a stochastic, population-based** evolutionary computer algorithm

- based on **social-psychological principles** and provides insights into social behavior, as well as contributing to engineering applications.

PSO first described in 1995 by James Kennedy and Russell C. Eberhart.

Techniques have evolved greatly, and the original version of the algorithm is barely recognizable in the current ones.

PSO is like GAs

Particle swarm optimization (PSO) is a

population based stochastic optimization technique

inspired by social behavior of **bird flocking** or **fish schooling**.

PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA).

initialized with a **population of random solutions**

searches for optima by **updating generations**.

PSO is not like GAs

PSO has **no evolution operators** such as crossover and mutation.

In PSO, the potential solutions, called **particles, fly through the problem space** by following the current optimum particles.

The advantages of PSO are

- easy to implement and
- there are few parameters to adjust.

PSO has been successfully applied in many areas: function optimization, artificial neural network training, fuzzy system control, and other areas where GA can be applied.

Social Learning

Social influence and social learning enable a person to maintain cognitive consistency.

People solve problems by

- **talking with other people** about them, and
- as they **interact** their beliefs, attitudes, and behaviors **change**;
- the changes could typically be depicted as the **individuals moving toward one another** in a sociocognitive space.

Particle Swarm

The particle swarm simulates this kind of social optimization.

A problem is given, and a way to evaluate a proposed solution
  the **fitness function**.

A **communication structure** or social network is also defined
  neighbors for each individual

Then a **population of individuals** defined
  random guesses at the problem solution
  These individuals are candidate solutions.
  They are also known as the particles, hence the name particle swarm.

The Algorithm

An **iterative process** to improve these candidate solutions

The particles

- iteratively **evaluate the fitness** of the candidate solutions
- And **remember the location** where they had their best success.

The individual's best solution is called the **particle best** or the **local best**.

Each particle

- makes this information available to their **neighbors**.
- And can see where their neighbors have had success.

Movements through the search space are guided by these successes, with the **population converging at the end of a trial**

- Not necessarily on a global optimum

Particles

The swarm is modeled by particles in multidimensional space that have

1. a position and
2. a velocity.

These particles fly through hyperspace (i.e., $\Re^n$) and have two essential reasoning capabilities:

their memory of their **own best position** and

knowledge of the **global or their neighborhood's best**.

In a minimization optimization problem, "best" simply means the position with the smallest objective value.

Members of a swarm

1. **communicate good positions** to each other and
2. **adjust their own position and velocity** based on these good positions.

Particle Information

So a particle has the following information to make a suitable change in its position and velocity:

A **global best** that is known to all and immediately updated when a new best position is found by any particle in the swarm.

Neighborhood best that the particle obtains by communicating with a subset of the swarm. - sometimes

The **local best**, which is the best solution that the particle has seen.

Particle Position

The particle position and velocity update equations in the simplest form that govern the PSO are given by

$$
\begin{aligned}
v_{i,j} \leftarrow c_0 v_{i\,j} + {} & \\
& c_1 r_1 (globalbest_j - x_{i,j}) + \\
& c_2 r_2 (localbest_{i,j} - x_{i,j}) + \\
& c_3 r_3 (neighborhoodbest_j - x_{i,j})
\end{aligned}
$$

$$x_{i,j} \leftarrow x_{i,j} + v_{i,j}$$

i is the particle and *j* is the dimension

Convergence

As the swarm iterates, the fitness of the global best solution improves - decreases for minimization problem

All particles being influenced by the global best **eventually approach the global best**, and the fitness may never improve despite however many runs the PSO is iterated thereafter.

Particles move about in the search space in close proximity to the global best - **not exploring the rest of search space**.

This phenomenon is called '**convergence**'.

Is Convergence good?

- Yes and no
  - Early convergence
  - Convergence on global optima

Coefficients

If the inertial coefficient of the velocity is small,

all particles could slow down until they approach zero velocity at the global best.

The **selection of coefficients** in the velocity update equations affects the convergence and the ability of the swarm to find the optimum.

You can **reinitialize the particles positions** at intervals or when convergence is detected.

Neighborhood Bests

The next algorithm uses the global best and local bests but no neighborhood bests.

Neighborhood bests

- allow **parallel exploration** of the search space
- **reduce** the susceptibility of PSO to falling into **local minima**
- **slow down convergence speed**.

do not create isolated subswarms because of **overlapping of neighborhoods**:

- to make neighborhoods of size 4, say,
  - particle 1 would only communicate with particles 2 through 5,
  - particle 2 with 3 through 6, and so on.

Use of Neighborhoods

a new best position discovered by particle 2's neighborhood would be communicated to particle 1's neighborhood at the next iteration of PSO

Smaller neighborhoods lead to **slower convergence**

larger neighborhoods to **faster convergence**

global best representing a neighborhood consisting of the entire swarm.

Neighborhoods are not used much anymore

The Power

A single particle by itself is unable to accomplish anything.

The power is in interactive collaboration.

Formally

Let $f: \Re^m \rightarrow \Re$ be the fitness function that takes a particle's solution with several components in higher dimensional space and maps it to a single dimension metric.

Let there be n particles, each with associated positions $x_i \in \Re^m$ and velocities $v_i \in \Re^m$, $i=1,\ldots,n$.

Let $\hat{x}_i$ be the current best position of each particle and let $\hat{g}$ be the global best.

Initialize

Initialize x_i and v_i for all i.

Take $x_{ij} \in U[a_j,b_j]$ and $v_i=0$ for all i and $j=1,\ldots,m$, where a_j,b_j are the limits of the search domain in each dimension, and U represents the Uniform distribution (continuous).

and

$\hat{x}_i \leftarrow x_i$ current best position

$\hat{g} \leftarrow \arg\min_{x_i} f(x_i), i = 1,\ldots,n$ global best position

Code

```
While not converged:
  For each particle :
        Create random vectors $r_1,r_2$ : $r_{1j}$ and $r_{2j}$ for all $j$, by
  taking $r_{1j}, r_{2j} \in U[0,1]$ for $j=1,\ldots,m$
        Update the particle positions: $x_i \leftarrow x_i + v_i$
        Update the particle velocities: $v_i \leftarrow \omega v_i + c_1 r_1 \circ (\hat{x}_i - x_i) + c_2 r_2 \circ (\hat{g} - x_i)$
        Update the local bests: $if\ f(x_i) < f(\hat{x}_i), \hat{x}_i \leftarrow x_i$
        Update the global best: $if\ f(x_i) < f(\hat{g}), \hat{g} \leftarrow x_i$
$\hat{g}$ is the optimal solution with fitness $f(\hat{g})$
```

Parameters

ω is an **inertial constant** usually slightly less than 1.

c_1 and c_2 are constants.

represent a "**cognitive**" and a "**social**" component,

affect how much the particle's personal best and the global best (respectively) influence its movement. Usually $c_1, c_2 \approx 2$.

r_1, r_2 are two **random vectors**

each component a uniform random number between 0 and 1
a **NEW random component per dimension**

o operator indicates element-by-element multiplication

the Hadamard matrix multiplication operator.

Intuition

- essentially carrying out a **discrete-time simulation** (each iteration of it represents a "tick" of time)
- particles "**communicate**" information they find about each other by **updating their velocities** in terms of local and global bests.
- When a new best is found, the **particles will change their positions** accordingly so that the new information is "broadcast" to the swarm.
- The particles are **always drawn** back both to their own **personal best** positions and also to **the best position of the entire swarm**.
- They also have **stochastic exploration** capability via the use of the random multipliers r_1,r_2.

Modifications

- clamp the velocities to a certain maximum amount
- a probabilistic bit-flipping local search heuristic to the loop
- frequently the value of ω is taken to decrease over time; e.g., one might have the PSO run for a certain number of iterations and DECREASE linearly from a starting value (0.9, say) to a final value (0.4, say) in order to facilitate exploitation over exploration in later states of the search

Local Maxima

- a stochastic hill-climber risks getting stuck at local maxima, but the stochastic **exploration and communication** of the swarm overcomes this.
- as "niching" versions designed to find **multiple solutions** to problems where it is believed or known that there are multiple global minima which ought to be located

Evolutionary Techniques

Most of evolutionary techniques have the following procedure:

1. Random generation of an initial population
2. Reckoning of a fitness value for each subject.
3. Reproduction of the population based on fitness values.
4. If requirements are met, then stop. Otherwise go back to 2.

Parameter Tuning

The number of particles: the typical range is 20 - 40. Actually for most of the problems 10 particles is large enough to get good results. For some difficult or special problems, one can try 100 or 200 particles as well.

Dimension of particles: It is determined by the problem to be optimized,

Range of particles: It is also determined by the problem to be optimized, you can specify different ranges for different dimension of particles.

Vmax: it determines the maximum change one particle can take during one iteration. Usually we set the range of the particle as the Vmax for example, the particle (x1, x2, x3) if x1 belongs [-10, 10], then Vmax = 20

Learning factors: c1 and c2 usually equal to 2. However, other settings were also used in different papers. But usually c1 equals to c2 and ranges from [0, 4]

More Parameter Tuning

The stop condition: the maximum number of iterations the PSO execute and/or the minimum error requirement. this stop condition depends on the problem to be optimized.

Global version vs. local version: we introduced two versions of PSO. global and local version. global version is faster but might converge to local optimum for some problems. local version is a little bit slower but not easy to be trapped into local optimum. One can use global version to get quick result and use local version to refine the search.

Another factor is inertia weight, which is introduced by Shi and Eberhart

Similarities

PSO shares many common points with GA.

- Both start with a **randomly generated population**
- Both have **fitness values** to evaluate the population.
- Both update the population and search for the optimium with **random techniques**.

Both systems do not guarantee success.

Differences

PSO does **not** have **genetic operators** like crossover and mutation.

Particles **update themselves** with the internal velocity.

They also have **memory**, which is important to the algorithm.

Compared with genetic algorithms (GAs), the information sharing mechanism in PSO is significantly different.

One Way Information Sharing

In GAs, chromosomes share information with each other (or at least all their children)

The whole population moves like one group towards an optimal area.

In PSO, only gBest (or lBest) gives out information to others.

It is a **one-way information sharing** mechanism.

The evolution only looks for the best solution.

Compared with GA, all the particles tend to converge to the best solution quickly even in the local version in most cases.

Why this isn't Right

In GAs only fit individuals will pass on information

It is true that everyone has a chance to pass on information, but certainly not all the genes do

Summary

PSO will converge faster

PSO will always converge

It will only give a single solution

It won’t necessarily be the global optima

Shafiq’s Phd

Uses PSO for clustering

- Starts with many clusters - particles
- One cluster/particle eats the closest piece of data
- New Centroid of cluster is defined
- Repeat

Pick the level with the best inter/intra cluster distance

Shafi’s Results

Better results then k-means

Don’t need to know k beforehand

Moved on to Outlier Recognition and Recommender Systems

END