Nature-Inspired Computing

Particle Swarm Optimization

Dr. Şima Uyar
September 2006

Introduction

- universal behavior of individuals leads to cultural adaptation:
  - evaluate
  - compare
  - imitate
- these can be modeled in computer programs to solve hard problems

Evaluate

- tendency to evaluate stimuli as positive or negative / good or bad
- leads to learning
  - individual can learn to distinguish the features of the environment as good or bad

Compare

- individuals in a population compare themselves to others
- serves as a motivation to learn and change
- standards for social behavior set by comparing to others

Imitate

- imitating a behavior which has led an individual to be superior

PSO

- population based
- stochastic
- developed by Eberhart, Kennedy, 1995
- inspired by social behavior of bird flocking or fish schooling

PSO

- random initial population
- search for optima through updating generations
- potential solutions: particles
- particles move through system following the current optimum particles

PSO

- two variations
  - binary PSO
  - continuous (real-valued) PSO
    - more popular version

PSO Applications

- function optimization
- ANN training
- fuzzy system control
- similar areas EAs are applied to

PSO

- each particle (solution)
  - has fitness value (objective function)
  - has velocity (directing flight)
  - has position
- each particle made up of
  - string of binary decision variables – binary PSO
  - vector of real-valued variables – continuous PSO

PSO

- available info for each particle
  - its own experience
  - experience of those around it
- particles follow two current best
  - pbest: best solution it has achieved so far
  - gbest: best solution any particle has achieved so far (global)
    - if only topological neighbors considered: lbest

```
for each particle
  initialize particle;
do {
  for each particle {
    calculate fitness;
    if fitness > pbest
      set current value as pbest;
  }
  choose gbest;
  for each particle {
    calculate particle velocity;
    update particle position;
  }
} while max. no of iterations or min.
      error ctiteria not met;
```

!! velocities in each dimension have a V_{max} !!

Continuous PSO

- particle velocity update rule:

$$v[] = v[] + c1 * rnd * (pbest[] - present[]) + c2 * rnd * (gbest[] - present[])$$
$$present[] = present + v[]$$

where
- v[]: particle velocity
- present[]: current best solution
- rnd: random number in [0,1]
- c1 and c2: learning factors
  - usually c1=c2=2 or c1+c2=4

Continuous PSO

- V_{max}
  - determines max change amount for each particle for one iteration
  - usually range is used
  - e.g. if x is in [-10,10] V_{max} is 20

Binary PSO

- an individual needs to make a set of decisions based on
  - its past experience
  - inputs from the social environment
  - its current position regarding issue

Binary PSO

Probability of an individual's answer:

$$P(x_{id}(t) = 1) = f(x_{id}(t-1), v_{id}(t-1), p_{id}, p_{gd})$$

where

$P(x_{id}(t) = 1)$	is the probability that individual i will choose 1 for d*th* bit
$x_{id}(t)$	is the current state of individual i for d*th* bit
$v_{id}(t-1)$	is the individual i's current prob. of choosing 1 for d*th* bit
p_{id}	is the individual i's best state found so far for d*th* bit
p_{gd}	is the neighborhood best state found so far for d*th* bit

Binary PSO

$v_{id}(t-1)$
- shows individuals predisposition to choose 1
- determines a probability threshold
- for higher values, individual more likely to choose 1
- has to be between 0 and 1
- want to adjust individual's disposition towards its success and that of the community
- will be updated at eeach step

Binary PSO

$$v_{id}(t) = v_{id}(t-1) + \varphi_1(p_{id} - x_{id}(t-1)) + \varphi_2(p_{gd} - x_{id}(t-1))$$

$usually \quad (\varphi_1 + \varphi_2 = 4.0)$

$$if \quad \rho_{id} < s(v_{id}(t)) \quad then \quad x_{id}(t) = 1; \quad else \quad x_{id}(t) = 0$$

$$s(v_{id}) = \frac{1}{1 + \exp(-v_{id})}$$

ρ_{id} is a vector of uniformly distributed random numbers between 0.0 and 1.0

$V_{min} \le v_{id} \le V_{max}$

$usually\ chosen\ as\ V_{max} = \pm 4.0$

$so\ that\ s(V_{max}) \approx 0.018$ ⇐ smallest prob. of a bit changing

! V_{max} similar to mutation rate in EAs

PSO x GA

- both use random initial populations
- both use a fitness function to evaluate solution candidates
- both update populations through generations to search for optima
- both do not guarantee success

PSO x GA

- in PSO: no evolutionary operators
- in GA: selection, crossover, mutation operators

- in GA: chromosomes share info
- in PSO: only best share info but only one way

PSO x GA

- PSO has memory
- in PSO all particles converge quickly (possibly to local optima)
- PSO is easier to implement
- PSO has fewer parameters to tune
- PSO (most commonly) uses real numbers for representing particles

Applying PSO

- when applying PSO determine
  - representation of solution
  - parameter settings
  - fitness function

Parameters in PSO

- no of particles
  - typically 20-40
  - sometimes 10 enough
  - for hard problems try 100-200

Parameters in PSO

- dimension of particles
  - depends on problem
- range of particles
  - depends on problem
  - dimensions may have different ranges

Parameters in PSO

- c1 and c2
  - learning factors
  - typically c1=c2=2
  - in some studies c1=c2 and in [0,4]

Parameters in PSO

- stopping condition
  - max no of iterations
  - min error requirement
- global x local versions
  - global is faster but possibly converges to local optima
  - possible to use global for getting a solution quickly and local for refining