Topic 14 – Particle Swarm Optimization

Friday, April 22 (Continued)

Each particle will have a velocity, that depends on three things:

1) its current velocity
2) the best solution that particle has ever seen
3) the best solution any particle has ever seen

vectors

Let $x_i(t)$ and $v_i(t)$ denote the position and velocity of particle i at time t.

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

(the velocity determines how the particle moves from one time to the next)

$$v_i(t+1) = \alpha \cdot v_i(t) + \beta \cdot r_1 \cdot \overbrace{\left(b_i(t) - x_i(t)\right)}^{\text{the vector from pos. to best sol.}}$$

$$+ \gamma \cdot r_2 \cdot \overbrace{\left(B(t) - x_i(t)\right)}$$

$b_i(t)$ = best solution particle i has seen by time t

$B(t)$ = best solution any particle has seen by time t

α, β, γ: weighting factors (fixed real #s) that you decide on ahead of time

Standard first try: $\alpha = 0.9$, $\beta = 1$, $\gamma = 1$

r_1 and r_2: random vectors in $[0,1]$.

Note: $b_i - x_i$ and $B - x_i$ are differences of solutions in the search space, so we need to have a definition of that.

Easy for continuous spaces ($\mathbb{R}^n$)

Demos:

→ Talk α, β, γ

Problem: What if your particles run away?

* You need to keep your particles in regions that satisfy the constraints.

* What do you do if your particle moves to an invalid spot?

Option 1) If a new position violates a constraint, just don't move. If you wait long enough, inertia decays ($\alpha < 1$), so eventually, you might move somewhere good.

Option 2) Destroy the particle and create a new one at a random position.

* One way the reduce the frequency of this happening is to declare a max. speed that the particles can have. If the speed is too high, we scale the velocity vector down.

* Sometimes it's helpful to add another term to velocity, in between "local best" and "global best". For every particle, randomly pick a few other particles (at the beginning) to be "informants".

Add a term to $v_i(t+1)$:

$$r_3 \cdot \delta \cdot \left([\text{best solution any of particle } i\text{'s informants have ever seen}] - x_i(t) \right)$$

* Note that there is nothing like H-C here. How could we incorporate some kind of H-C?

One way:
(1) Move all the particles.
(2) Make every individual particle hill climb for a while
(3) repeat

Topic 15 - Neighborhoods in Continuous Space

In our MHs so far that need a continuous tweak, we we've used is a very simple one: Start with a point

$$x = (x_1, x_2, \ldots, x_d)$$

$$s = \text{tweak}(x) = x + (r_1\delta_1, r_2\delta_2, \ldots, r_d\delta_d),$$

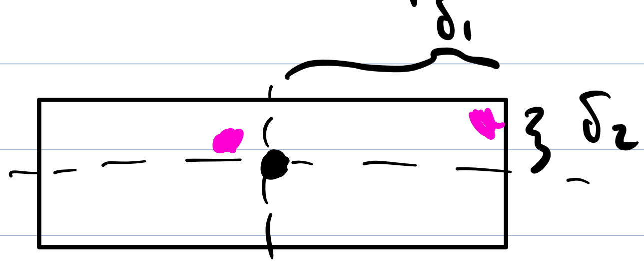
where each r_i is a uniform random

between -1 and 1 and δ_i is a predetermined parameter that specifies the maximum change allowed in that dimension.

* In most of our examples, the x and y bounds were the same, so we used
  $\delta_1 = \delta_2 = 0.05$

* With the spring example, we had
  $\delta_1 = \delta_2 = 0.01$, $\delta_3 = 0.1$

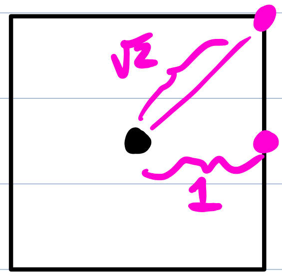
→ Randomly picking a point in a rectangle around the current point.



For now, assume $x = \vec{0}$, and $\delta_i = 1$ for all i.

$s = \text{tweak}(x) = (r_1, r_2, \dots, r_d)$

The new point s is somewhere in the d-dimensional cube with side length 2 centered at the origin.



What is the distance from the center of a d-dim. cube to a corner?

$(0,0,0,\ldots,0) \to (1,1,1,\ldots,1)$

$\sqrt{d}$