Monday, April 12

Announcements:
→ HW 5 due Wednesday

Topic 15 - Particle Swarm Optimization

In all of our MH so far: we have always tracked a single solution moving through the search space.

Particle Swarm Opt. (PSO) is our first "population MH" - we will track many candidates at a time and they will interact with each other.

Idea: You have N particles, each representing a candidate in the search space. They all start at random positions.

random candidates.

Each particle has a velocity that depends on three things:

(1) current velocity (inertia)
(2) the best solution that particle has ever seen before
(3) the best solution that any particle has ever seen before.

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

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

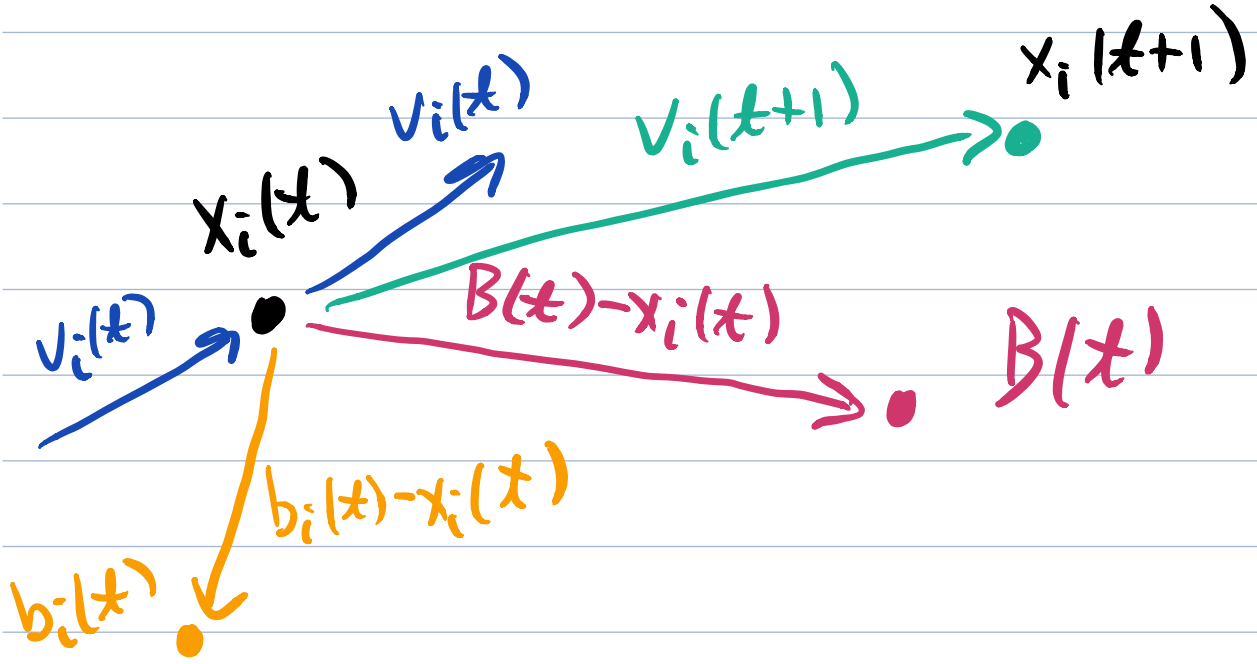
(the velocity dictates how the position changes)

$$v_i(t+1) = \underbrace{\alpha \cdot v_i(t)}_{\text{inertia}} + \beta \cdot r_1 \cdot \overbrace{\left(b_i(t) - x_i(t)\right)}^{\text{diff. b/w now and personal best}} + \gamma \cdot r_2 \cdot \overbrace{\left(B(t) - x_i(t)\right)}^{\text{diff b/w now and global best}}$$

α, β, γ: parameters
r_1 and r_2: random vectors in $[0,1]$
$b_i(t)$: best sol. particle i has ever seen up to time t
$B(t)$: best sol any particle has seen

up to time t.

α: weighting factor for inertia ≈ 0.9
β, γ: weighting factors for personal / global best ≈ 1



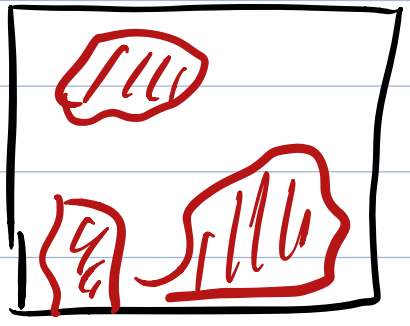
We need to have a meaning for subtracting solutions in the search space.

Continuous optimation: $\mathbb{R}^d$

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} - \begin{bmatrix} d \\ e \\ f \end{bmatrix} = \begin{bmatrix} a-d \\ b-e \\ c-f \end{bmatrix}$$

Problem: What do you do if your particles run away?

* You need to keep your particles in regions that satisfy the constraints.
  -> Could be nice bounds, $-2\pi \leq x, y \leq 2\pi$, or worse like in the spring problem.



-> What do you do if your particle moves into an invalid spot?