• Update files for Assignment 3:
  • Particle.java.
  • Fitness values for assignment 3-2.

Binary Representation
• Applications using binary representation:
  – Select the genes from the patients’ micro-array data that contributes to a disease.
  • 10 genes example: 1 0 1 1 0 0 1 0 1 1
  – Encode real-valued optimization problem
  • 5 bits (0-31) for integer, 7 bits (0-127) for decimal
  • Example: 10001000001 00001011100 00110000000
  • Decoding: 17.1 1.5 6.0

Example
• In feature selection:
  – 0 1 1 1 0: select 2nd, 3rd and 4th genes
  – 1 0 0 0 1: select 1st and 5th genes
• In numerical (real value) optimization problem
  – 0 1 1 1 0 1111110: 14.126
  – 1 0 0 0 1 0000001: 17.1

Binary Search Space
• Binary Search Space:
  – A particle moves to nearer and farther corners of the hypercube by flipping various numbers of bits.
  – A smaller number of bit flips keep the particle in the near neighborhood.
  – When a particle reverses all of its binary coordinates from 0 to 1 and from 1 to 0, it moves to the farthest position from the current position in the binary search space.

Velocity in Binary PSO
• Velocity $v_{id}$ is the probability of bit $x_{id}$ taking the value of 1.
  – E.g. $v_{id}=0.2$, the probability that $x_{id}=1$ is 20% and the probability $x_{id}=0$ is 80%.
  \[ v_{id} = v_{id}(t-1) + c_1 (p_{best}(t-1) - x_{id}(t-1)) + c_2 (g_{best}(t-1) - x_{id}(t-1)) \]
  – A pbest bit ($p_{id}$) can be either 0 or 1.
  – A current $x_{id}$ can be either 0 or 1.
  – ($p_{id}$, $x_{id}$) can be 0, -1, 1.
  – The calculated $v_{id}(t)$ can be positive, negative or > 1
Example

• $x(t-1) = (0,1,1,1,0); v(t-1) = (-0.3,0.6,-0.8,0.1,0.5)$
• $pbest(t-1) = (1,0,0,1,1); gbest(t-1) = (1,1,1,1,1)$
• $v(t) = v(t-1) + c_1 r_1 (pbest(t-1) - x(t-1)) + c_2 r_2 (gbest(t-1) - x(t-1))$
• $v(t) = (-0.3,0.6,-0.8,0.1,0.5) + c_1 r_1 ((1,0,0,1,1) - (0,1,1,1,0)) + c_2 r_2 ((1,1,1,1,1) - (0,1,1,1,0))$
• $v(t) = (-0.3,0.6,-0.8,0.1,0.5) + c_1 r_1 (1,-1,-1,0,1) + c_2 r_2 (1,0,0,0,1)$

Velocity in Binary PSO - Continue

• We can normalize the original velocity to be a value between 1 and 0 using a sigmoid function:

$$v'_i(t) = \frac{1}{1 + e^{-v_i(t)}}$$

Particle Position Update

• No memory:
  - $x_{id}(t-1)$ is not used to update $x_{id}(t)$.
• $r_{id}$ is a random number between 0 and 1
  $$r_{id}(t) = \begin{cases} 1, & r_{id} < \text{sig}(v_{id}(t)) \\ 0, & \text{otherwise} \end{cases}$$
• If $x_{id}(t-1) = 0$, the probability of $x_{id}(t) = 1$ is $\text{sig}(v_{id}(t))$
• If $x_{id}(t-1) = 1$, the probability of $x_{id}(t) = 0$ is $(1 - \text{sig}(v_{id}(t)))$
• The probability of bit change is $P(\Delta) = \text{sig}(v_{id}(t))\{1 - \text{sig}(v_{id}(t))\}$

Velocity in PSO search

• Smaller velocity, approaching to 0, gives a larger $p(\Delta)$, hence more exploration.
• This is different from continuous PSO where a larger velocity gives bigger jump, hence more exploration.
• When $v_{id}(t)$ is 0, $\text{sig}(v_{id}(t)) = 0.5$, the $P(x_{id}(t) = 0) = 0.5$, which leads to random search.

Vmax in Binary PSO

• The maximum velocity $V_{\text{max}}$ limits the velocity values:
  - $|v_{id}| < V_{\text{max}}$
• It limits the probability that $x_{id}$ will take on a 0 or 1 value:
  - $V_{\text{max}}=0$, $\text{sig}(v_{id}(t))=0.5$, which gives random search.
• When a swarm is converged, the updated velocity is close to $V_{\text{max}}$:
  $$v_i(t) = v_i(t-1) + c_1 r_1 (pbest(t-1) - x_i(t-1)) + c_2 r_2 (gbest(t-1) - x_i(t-1))$$
  - $V_{\text{max}}=6$, $0.0025 < \text{sig}(v_{id}(t)) < 0.9975$, $P(\Delta) = \text{sig}(v_{id}(t))\{1 - \text{sig}(v_{id}(t))\}$
  - $\Rightarrow P(\Delta) < 0.000246$.
• $V_{\text{max}}=10$, $0 < \text{sig}(v_{id}(t)) < 1$, $P(\Delta) = \text{sig}(v_{id}(t))\{1 - \text{sig}(v_{id}(t))\} = 0$.
  - Particle position updates are less likely to happen.

Vmax in Binary PSO - Continue

• In other words, Vmax controls the further exploration after the swarm population has converged (can be premature converged).
• Note that unlike continuous PSO where a larger Vmax allows bigger jump, hence more exploration, smaller Vmax in binary PSO allows bigger jump, hence more exploration.
Exercise

• 6-bit Trap function:

\[ f(x) = \begin{cases} 3 & \text{if } x < 4 \\ 9 & \text{otherwise} \end{cases} \]

where \( c \) is the number of 1's in the particle solution.

\[ f(x) = 3 \times (4 - c) \times c \times (4 - c), \quad c < 4 \]

\[ 9 \times (4 - c) \times c, \quad \text{otherwise} \]

Exercise

\[
\begin{align*}
W &= 0 \\
W &= 1 \\
W &= 0
\end{align*}
\]

<table>
<thead>
<tr>
<th>i</th>
<th>solution</th>
<th>t=0</th>
<th>velocity</th>
<th>fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0 1 0 0 0 0</td>
<td>0 0 0 0 0</td>
<td>0 0 0 0 12</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>0 0 1 0 0 0</td>
<td>0 1 -1 -1 -1</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>0 0 1 0 0 0</td>
<td>0 1 -1 -1 -1</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>0 0 0 0 0 0</td>
<td>0 0 0 0 0</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>0 1 0 0 0 0</td>
<td>-1 0 0 -1 -1</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>0 1 0 0 0 0</td>
<td>-1 1 0 0 -1</td>
<td>0 0 0 0</td>
<td></td>
</tr>
</tbody>
</table>

Swarm population converged to all 0

Exercise

<table>
<thead>
<tr>
<th>i</th>
<th>solution</th>
<th>t=1</th>
<th>velocity</th>
<th>fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0 0 0 0 0 0</td>
<td>0 0 0 0 0</td>
<td>0 0 0 0 12</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>0 1 0 0 0 0</td>
<td>0 1 -1 -1 -1</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>0 1 0 0 0 0</td>
<td>0 1 -1 -1 -1</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>0 0 0 0 0 0</td>
<td>0 0 0 0 0</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>0 0 0 0 0 0</td>
<td>-1 1 0 0 -1</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>0 1 0 0 0 0</td>
<td>-1 1 0 0 -1</td>
<td>0 0 0 0</td>
<td></td>
</tr>
</tbody>
</table>

Inertia weight in Binary PSO

• When \( w < 1 \), the binary PSO won’t converge.
  - When \(-1 < w < 1\), \( v_{ij} \) becomes 0 over time (\( \text{sig}(v_{ij}(t+1)) = 0.5 \)). This gives pure random search.

• When \( w > 1 \), \( v_{ij} \) increases over time (\( \text{sig}(v_{ij}(t+1)) = 1 \)).
  - All bits become 1.

• When \( w < -1 \), \( v_{ij} \) decrease over time (\( \text{sig}(v_{ij}(t+1)) = 0 \)).
  - All bits become 0.