Review the rest

Selection Based on Scaling and Ranking Mechanisms

- Static Scaling Mechanisms
  - linear scaling

$$s(x) = ax + b$$

$$tf(x) = s(f(x))$$

Evaluation function: $eval(x) = tf(x) = s(f(x)) = af(x) + b$

  - Power law scaling

$$eval(x) = tf(x) = s(f(x)) = (f(x))^u$$

- Logarithmic scaling

$$tf(x) = S(f(x)) = e^{\frac{f(x)}{T}}$$

T is controlling the selective pressure →>decrease during search process

Selection pressure→degree to which highly fit individuals are allowed to produce offspring (copies) in the next generation

Low values of selection pressure→ provide each individual in the population with a reasonable selection prob.

High values of selection pressure→strongly favor the best individuals in the population→ population diversity decrease quickly

- Dynamic Scaling mechanism
  - Sigma truncation

$$T(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{else} \end{cases}$$

$$S(x) = T\left(x - \left(m_t - c\sigma_t\right)\right)$$

$$eval(x) = tf(x) = S\left(f(x)\right) = T\left(f(x) - \left(m_t - c\sigma_t\right)\right)$$

$$eval(x) = \begin{cases} f(x) - \left(m_t - c\sigma_t\right) & \text{if } f(x) > \left(m_t - c\sigma_t\right) \\ 0 & \text{if } f(x) \le \left(m_t - c\sigma_t\right) \end{cases}$$

OR

$$eval(x) = \begin{cases} 1 + \dfrac{f(x) - m_t}{\sigma_t} & \text{if } \sigma_t \neq 0 \\ 0 & \text{if } \sigma_t = 0 \end{cases}$$

- Window scaling

$$eval(x) = f(x) - \min_{y \in P(t)} f(y)$$

OR

$$eval(x) = f(x) - \min_{y \in W} f(y)$$

where W is the set of all individuals contained in the last g generation, $g \geq 1$

- Fitness remapping for minimization problem

$$tf(x) = M - f(x)$$ where M is the upperbound of the objective function

$$eval(x(t)) = tf(x(t)) = M_t - f(x(t))$$ where $x(t) \in P(t)$ and M_t is max value of function f found so far or max value of f found within generation $P(t)$

<u>OR</u> $$eval(x(t)) = \frac{1}{K + f(x(t)) - f_{\min}(t)}$$ where K is positive constant and $f_{min}(t)$ is the minimum observed value of function f up to time t

<u>OR</u> $$eval(x) = \frac{1}{K + f(x)}$$

- Rank-based Selection

  let number of individuals in each generation → N ($|P(t)|=N$)

  - Linear Ranking selection

    selection probability of x^i

$$p_i = \frac{1}{N}\left[Min + (Max - Min)\frac{r_i - 1}{N-1} \right]$$

    where r_i → rank of individual x^i (ranked in increasing order)

    expected value of offspring of x^i

$$n_i = Np_i$$

And $\sum_{i=1}^{N} n_i = \sum_{i=1}^{N} Np_i = N$ Then $\sum_{i=1}^{N} Np_i = N$

$$\sum_{i=1}^{N} N\frac{1}{N}\left[Min + (Max - Min)\frac{r_i - 1}{N-1} \right] = \sum_{i=1}^{N} Min + \sum_{i=1}^{N} (Max - Min)\frac{r_i - 1}{N-1} = N$$

$$NMin + \frac{1}{N-1}(Max - Min)\sum_{i=1}^{N}(r_i - 1) = N$$

But, since $\sum_{i=1}^{N}(r_i - 1) = \sum_{i=1}^{N}(i-1) = \sum_{i=0}^{N-1} i = \frac{N(N-1)}{2}$

Hence, $Min + \frac{1}{2}(Max - Min) = 1$

$Min = 2 - Max$ OR $Max + Min = 2$

Since $Min \geq 0$ and $Min \leq Max \rightarrow Max \leq 2$ and $Max \geq 1 \rightarrow 1 \leq Max \leq 2$

Hence $p_N = \frac{Max}{N}$ and $p_1 = \frac{2 - Max}{N} = \frac{Min}{N}$

Alternatively, $p_i = \frac{1}{N}\left[Max + 2(Max - 1)\left(1 - \frac{r_i - 1}{N-1}\right)\right]$

- nonlinear ranking selection

  Selection probability will be

$$p_i = \frac{1}{c}\left(1 - e^{1-r_i}\right)$$

- SUS Algorithm

1. *ptr* = Rand(); */*random within* [0,1] → *(uniform distribution)*/*
2. *For (sum=i=0; i<N;i++)*
   1. *For (sum +=n_i; sum>ptr; ptr++)*
   2. *Select(i);*
   3. *End For*
3. *End For*

guarantee that x^i will reproduce at least $\lfloor n_i \rfloor$ but not more than $\lceil n_i \rceil$

- Example

Chomosome	$f(x)=x^2$	r_1	p_i	n_i
$x^1=10$	100	1	$\frac{2-1.2}{3}=0.27$	3(0.27)=0.81
$x^2=15$	225	2	$\frac{1}{3}\left[0.8+(1.2-0.8)\frac{2-1}{3-1}\right]=0.33$	3(0.33)=0.99
$x^3=20$	400	3	0.4	3(0.4)=1.2

Suppose ptr is 0.8→ the selection will be x^1, x^3 and x^3 , respectively

- Tournament selection

  Similar to rank selection in terms of selection pressure → but more efficient

  - Binary tournament (deterministic)
    - 2 individuals directly compete for selection
    - With or without reinsertion of the competing individual into the original population
    - based on direct pairwise comparison of individuals
      - 2 chromosomes are chosen at random from the population
      - The fitness of the chosen chromosomes is calculated
      - The most successful chromosome (the winner) is selected→ to intermediate population
  - Probability tournament
    - Step 3→ random $R \in [0,1]$ if $R<p$ then the fitter of the two is selected; otherwise the less fit is selected

      where p is preselected or p varied with time →

$$p(t) = p_0 e^{-ct} \quad \text{or} \quad p(t) = p_0 \frac{c}{1+\ln t}$$ where $p_0 \in (0,1)$ and c is positive constant

- Boltzmann tournament

  x → current solution, y→ alternative solution →binary tournament between x and y with

$$p(x) = \frac{1}{1 + e^{\frac{f(x) - f(y)}{T}}}$$

  Where T is the temperature→ reduced in a predefined manner in successive iteration

  The winner of the trial will be the current solution in the next trial

- Elitism→ selection of a set of individual from the current generation to survive to the next generation

The fraction of new individual at each generation has been called generation gap

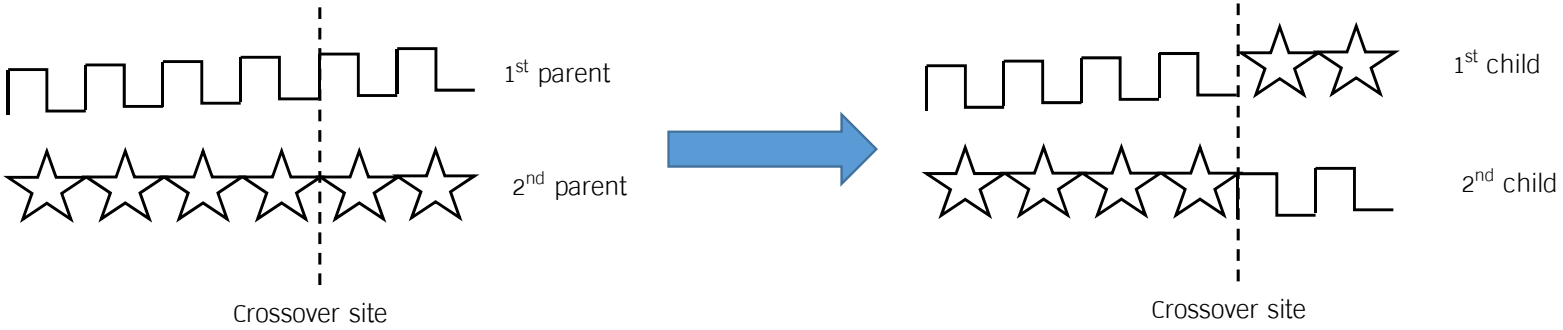
- Recombination operator

  cross over: $C: X \times X \rightarrow X \times X \quad \rightarrow \quad C(x,y)=(x',y')$

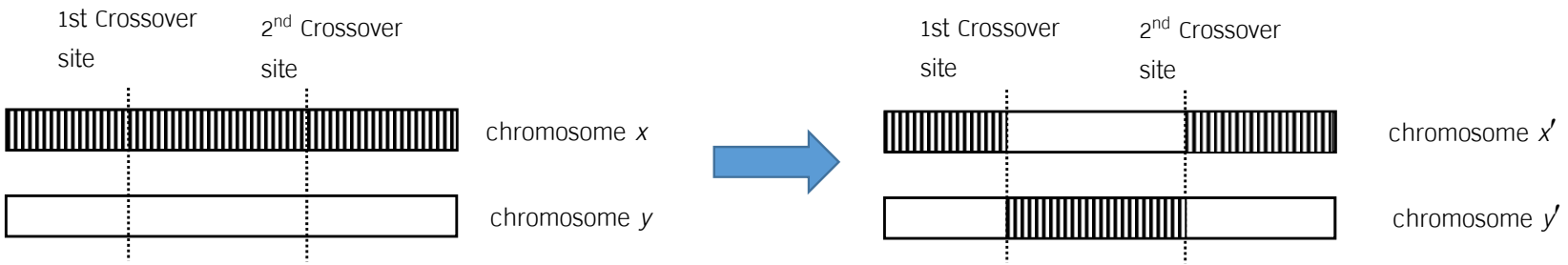
  - One-point crossover

  $x = x_1\, x_2\, x_3 \ldots\, x_L$ and $y = y_1\, y_2\, y_3 \ldots\, y_L$ after cross over

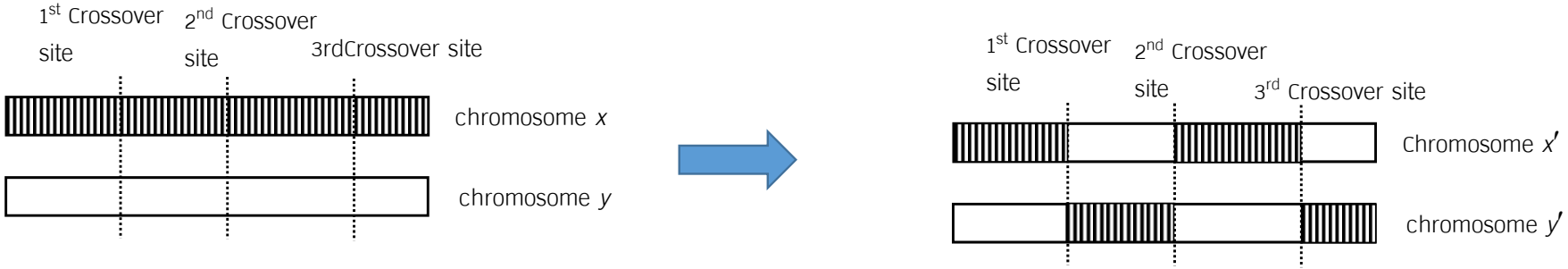
  $x' = x_1\, x_2\, x_3 \ldots\, x_{k-1}\, x_k\, y_{k+1} \ldots\, y_L$ and $y' = y_1\, y_2\, y_3 \ldots\, y_{k-1}\, y_k\, x_{k+1} \ldots\, x_L$



- Two-point cross over



- *N*-point cross over



- *N*-point cross over algorithm

P1: for each chromosome from intermediate population (P^1)

P1.1: random $q \in [0,1]$ →uniform distribution

P1.2: if $q < p_c$ (where p_c is mating probability) then respective chromosome is considered mating →go to mating pool else not mating

P2: let m^* →number of chromosome selected at P1

if m^* is even then select pair chromosome (random)

if m^* is odd then discard or select chromosome from P^1

P3: Cross over is applied on pairs in P2

P3.1: for each pair, generate k ($\leq k \leq L$)

and random crossing site $k_1, k_2, \ldots, k_N$ for ($N \geq 2$)

P3.2: descendants obtained at P3.1 become potential members of $P(t+1)$ → candidate for mutation

P3.3: parents of newly generated chromosome are discarded from P^1

P3.4: chromosomes left in P^1 are added to $P(t+1)$

- Punctuated crossover
  - Record the crossover points in the chromosome
  - If a certain crossover point has generated a very poor offspring this crossover point will not be considered further
  - If the fitness of the descendant is good, the crossover point is maintained as active
  - Cross-over point self adjust according to the previous dynamics of the search process

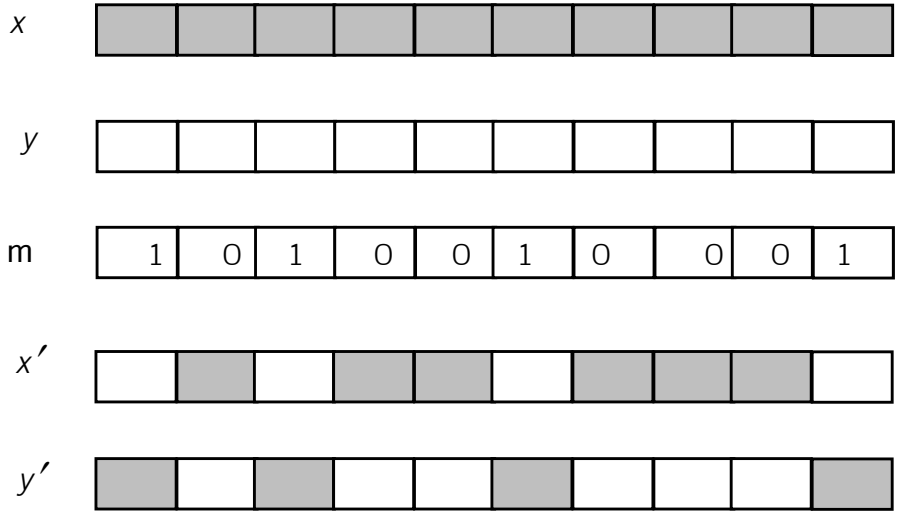
$C = x_1\, x_1 \ldots x_L k_1\, k_1 \ldots k_L$ when $k_i = 1$ → active crossover point, $k_i = 0$ → not a crossover point

During initialization probability of generating $k_i = 1$ is small

Set of crossover point used in the recombination of 2 chromosomes is the union of the parents

- Uniform crossover

1. *Set* $m_i = 0$ *for* $i = 1,2,\ldots,L$
2. *For each i*
   1. $random\ \xi \sim U(0,1)$
   2. *If* $\xi \leq p_{at\_i}$ *then* $m_i = 1$ *where* p_{at_i} *is the crossover probability at* i^{th} *position*



- Mutation

  Average number of position that will undergo a mutation→ $B = NLp_m$ where p_m →mutation probability ($p_m \in [0.001, 0.01]$), N→ number of chromosomes, L→length of chromosome

  Algo

  1. *For each chromosome และ For each position of the chromosome*
     1. *Generate a random number* q from $U(0,1)$
     2. *If* $q < p_m$ *invert that position*

        If $q \geq p_m$ *do not mutate*

  Weak mutation

  step 2 changed to if $q < p_m$ then 1 of 0 or 1 is chosen at random

- Non-uniform mutation

$$p_m(t) = p_m e^{-\beta t} \qquad \text{where } \beta \geq 1$$

OR

$$p_m(t) = \left(\frac{\alpha}{\delta}\right)^{\frac{1}{2}} \left(\frac{e^{\frac{-\beta t}{2}}}{L^{\frac{1}{2}} N} \right)$$

where $\alpha, \beta, \delta \rightarrow$ Real-valued positive parameter

OR

$$p_m(t) = \frac{1}{2 + \frac{L-2}{T} t}$$

where $0 \leq t \leq T \rightarrow p_m(0) = 0.5$ and $p_m(T) = \frac{1}{L}$

OR

$$p_m(x) = \frac{1}{2\left(f(x)+1\right) - L}$$

- Inversion operator

  Acts on a single chromosome. It inverts the values between 2 position of a chromosome. The 2 positions are chosen at random

- Simple genetic algorithm

1. *Set* $t = 0$

   Initialize ($P(0)$) → random

   a selection mechanism is chosen and if necessary a scaling mechanism

2. *The chromosome of* $P(t)$ are evaluated using the fitness function. The most successful individual from $P(t)$ is kept.

3. *Selection operator is applied* N *times*

   Selected chromosome make up an intermediate population P^1 (having N *members as well) representing candidate from the mating pool (MP). Some chromosomes from* $P(t)$ may have more copies in P^1 *while some others may have none*

4. *Crossover operator is applied to the chromosomes from MP*

   Newly generated chromosomes form P^2

   Parents of the chromosomes obtained through crossover are discarded from P^1

   Chromosomes that remained in P^1 become member of P^2

5. *Mutation operator is applied to* P^2 *outcome is the new generation*($P(t+1)$)

6. *set* $t = t+1$

   if $t < T$ (T is the maximum number of generation) then go to step 2 otherwise stop

- Solution→best member of the last population or best individual in all generation
- Can involve inversion or other types of operators

- Example

$$\max_{x_1, x_2} f(x_1, x_2) \quad \text{where} \quad f(x_1, x_2) = \frac{1}{1 + x_1^2 + x_2^2}$$

chromosome	x_1	x_2	Fitness value
C_0	−1	2	0.167
C_1	−2	3	0.007
C_2	1.5	0.0	0.31
C_3	0.5	−1.0	0.44

C_3, C_3, C_2 and C_0 are selected to P^1

(C_3, C_2) and (C_3, C_0)

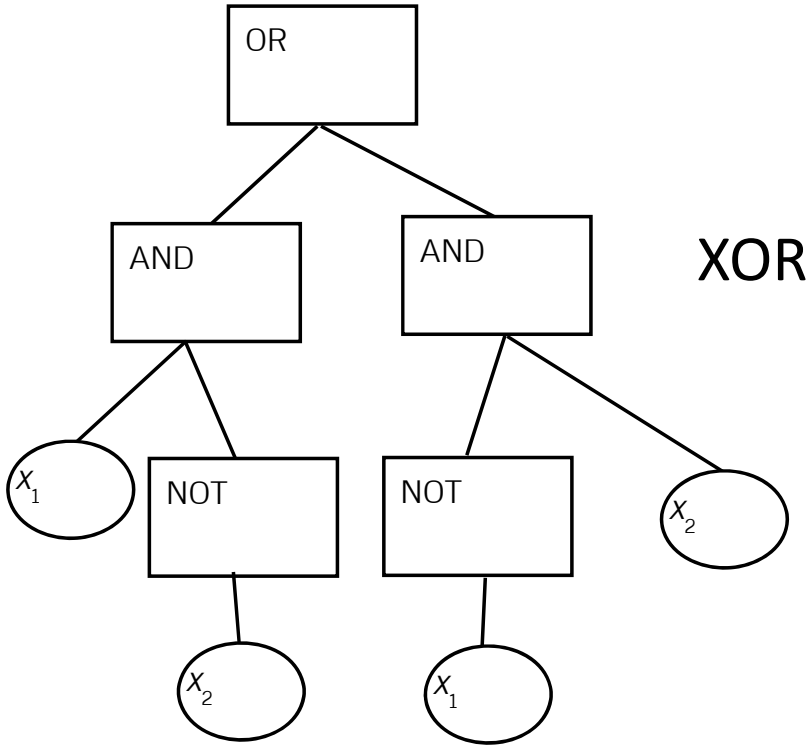
chromosome	x_1	x_2	Fitness value
C_0	0.5	0	0.8
C_1	1.5	−1.0	0.24
C_2	0.5	2.0	0.19
C_3	−1.0	−1.0	0.33

Genetic Programming

- Each individual/chromosome → 1 computer program represented using a tree structure
- Terminal set → specifies all variables and constant → elements of the terminal set form the leaf modes
- Function set → all functions that can be applied to the elements of the terminal set may be mathematical arithmetic and/or Boolean function or if-then-else→form the non-leaf nodes.

- Example

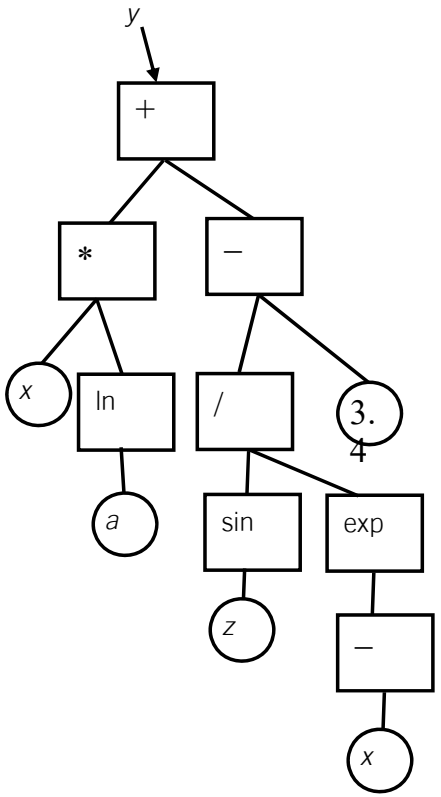
function $\{\text{AND, OR, NOT}\}$ and terminal set $\{x_1, x_2\}$ where $x_1, x_2 \in \{0,1\}$



XOR → (x_1 AND NOT x_2) OR (NOT x_1 AND x_2)

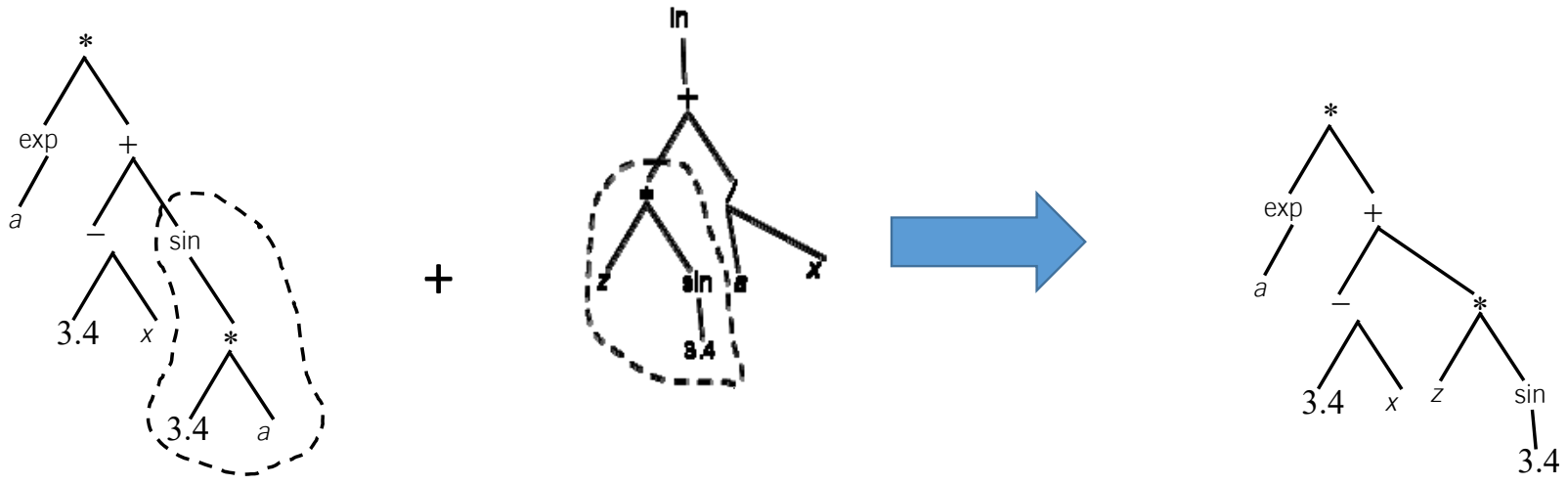
- Example

Function $\{-,+,*,/\}$ and terminal set $\{a,x,z,3.4\}$ where $a,x,z \in \Re$

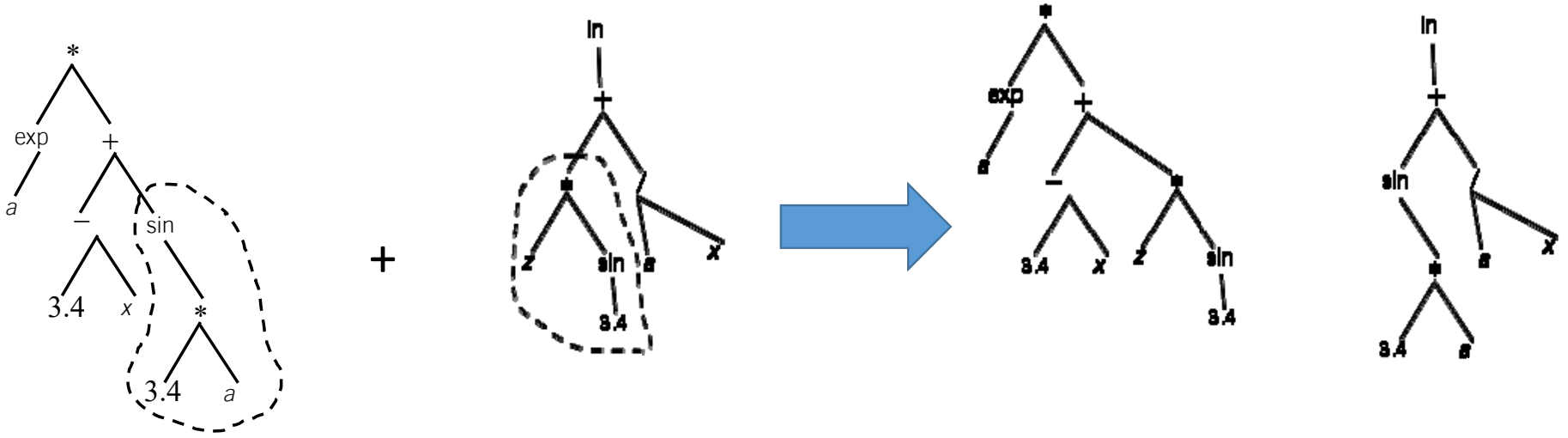


$$y = x * \ln(a) + \sin(z) / \exp(-x) - 3.4$$

Crossover→ 2 parents generating 1 child



Crossover→ 2 parents generating 2 children



- Mutation→choose a random point in a tree and replacing the subtree beneath that point by a randomly generated subtree
- Calculate the fitness of each program → by running it on a set of "fitness case"

Evolutionary programming

- Algo

1. *Let current generation be* $t=0$
2. *Initialize all parameters*
3. *Initialize population* $C(0)$ *with* n_s *individual*
4. *For each individual* $\mathbf{x}_i(t) \in C(t)$

   calculate fitness value $f(\mathbf{x}_i(t))$

   End
5. *While no convergence*
   - *For each individual* $\mathbf{x}_i(t) \in C(t)$

     mutate $\mathbf{x}_i(t)$ to produce offspring $\mathbf{x}_i'(t)$

     calculate fitness value $(f(\mathbf{x}_i'(t)))$

     add $\mathbf{x}_i'(t)$ *in offspring set* $C'(t)$

     End
   - *Select new population* $C(t+1)$ *from* $C(t) \cup C'(t)$ *using selection operator*
   - $t = t+1$

   End

Coevolution

- Predator-pray competitive coevolution→
  - Insects that eat plant
    - To survive plant needs to evolve mechanism to defend itself from the insects, and
    - The insects need the plants as food source to survive.
  - Inverse fitness interaction between 2 species→win for 1 species →failure for the other
- Symbiosis
  - Different species cooperate instead of compete→success in 1 species improve the survival strength of the other species→positive feedback among the species that take part in this cooperating process

- Coevolutonary algorithm (CoEA)
  - Competitive coevolution
    - Competition
    - Amensalism
  - Cooperative coevolution
    - Mutualism
    - Commensalism
    - Parasitism
- Competitive coevolution (CCE)
  - Evolve 2 population simultaneously
    - individual in 1 population → solution (evolve to solve as many test cases as possible)→ fitness $\propto$ the number of test cases solved by the solution
    - individual in another population → test case (evolve to present an incrementally increasing level of difficult to the solution individuals)→fitness is inversely proportional to the number of strategies that solve it

- Relative fitness function→express the performance of individual in one population is comparison with individuals in another population
  - Which individual from the competing population is used
  - Exactly how these competing individuals are used to compute the relative fitness
- Sampling scheme
  - All versub all sampling
  - Random sampling
  - Tournament sampling
  - All versus best sampling
  - Shared sampling
- Relative fitness function → $f(C_1.x_i)$
  - Simple function

  2 population → C_1 and C_2 →aim to calculate the relative fitness of each individual $C_1.x_i$ of C_1

  Sample of individuals is taken from C_2 → S

  sum $C_1.x_i$ = count the number of individuals in S that $C_1.x_i$ win

  $f(C_1.x_i)$ = sum $C_1.x_i$

- Fitness sharing

sim $C_1.x_i$ = similarities of $C_1.x_i$ with all the other individuals in that population → e.g., the number of individuals that also beats individual from C_2 samples

$$f\left(C_1.x_i\right) = \frac{sumC_1.x_i}{simC_1.x_i}$$

- Competitive fitness sharing

$$f\left(C_1.x_i\right) = \sum_{l=1}^{C_2.n_s} \frac{1}{C_1.n_l}$$

Where $C_2.x_1$, $C_2.x_2$,…, $C_2.x_{C2.ns}$,→ form C_2 and $C_1.n_l$ = total number of individuals in C_1 that defeat individual $C_2.x_l$

- Tournament fitness
  - Binary tournament determine a relative fitness ranking
  - Tournament fitness results in a tournament tree with root element as the best individual
    - Each level→ 2 opponent are randomly selected from that level and the best of the 2 advances to next level
  - If odd number→ a single individual from the level moves to the next level
  - After tournament→use any selection operator

- Hall of frame
  - At each generation the best individual of a population is stored in that population's hall of frames
  - May have limited size
  - New individual →inserted in hall of frame will replace the worst or the oldest individual
  - Individual from 1 population → complete against a sample of the current opponent population and its hall of frame
  - Prevents overspecialization

- Competitive coevolution with 2 population Algo
  - Initialize C_1 (solution), C_2 (test) population
  - While stopping condition(s) not true do
    - For each $C_1.x_i$ for i=1,…, $C_1.n_s$ do
      - Select a sample of opponents from C_2
      - Evaluate the relative fitness of $C_1.x_i$ w.r.t. this sample
    - End
    - For each $C_2.x_i$ for i=1,…, $C_2.n_s$ do
      - Select a sample of opponents from C_1
      - Evaluate the relative fitness of $C_2.x_i$ w.r.t. this sample
    - End
    - Evolve population C_1 for 1 generation
    - Evolve population C_2 for 1 generation
  - end
  - Select the best individual from C_1 as solution

- Competitive coevolution with 1 population Algo
  - Initialize 1 population *C*
  - While stopping condition(s) not true do
    - For each $C.x_i$ for i=1,…, $C.n_s$ do
      - Select a sample of opponents from *C* (exclude $C.x_i$)
      - Evaluate the relative fitness of $C.x_i$ w.r.t. the sample
    - End
    - Evolve population *C* for 1 generation
  - end
  - Select the best individual from *C* as solution

- Cooperative coevolution
  - Similar to divide and conquer
  - Only mutualism here
  - Individual from different species or subpopulation have to cooperate in some way to solve a global task
  - Fitness depends on that individual's ability to collaborate with individual from other species
  - Ex. For an n_x dimensional problem→n_x subpopulation (1 for each)
    - each subpopulation → responsible for optimizing one of the parameters of problem and no subpopulation can form a complete solution by itself
    - Merge representative from each subpopulation
    - Consider *j* subpopulation (C_j)→ each $C_j.x_i$ perform a single collaboration with the best individual from each other subpopulation → complete solution
    - Credit assign to $C_j.x_i$ ➔ fitness of the complete solution
    - Evolve independently from one another→ separate population is used to evolve each subcomponent using some EA

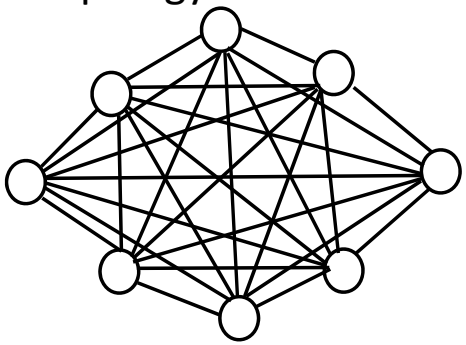
Swarm Intelligence (SI)

- Structured collection of interacting organisms (or agents)
- Individual organism
  - Simple in structure but their collective behavior can become complex
- Tight coupling between individual behavior and the behabior of the entire swarm
- Social interaction→direct (through visual audio, chemical context), or indirect (occur when one changes the environment and the other individuals respond to the new environment)
- 2 concepts
  - Self-organization
    - Positive feedback

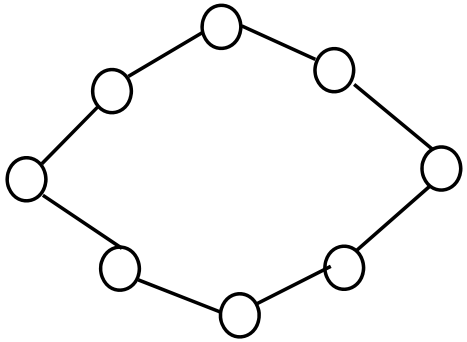
Swarm Intelligence

- 2 concepts
  - Self-organization
    - Positive feedback
    - Negative feedback
    - Fluctuation
    - Multiple interaction
  - Division of labor
- SI→no one is in charge, no one is giving orders, which the swarm individuals should obey or follow
- Social networkstructureof swarm→form an integral part of the existence of that swarm→provide the communication channel, able to self-organize to form the optimal nest structures

- Partical swarm optimization (PSO)
  - Simulation of social behavior of birds within flock
  - Particles are “flown” through hyperdimensional search space
  - Neighborhood topology
    - Star topology→first version PSO →gbest

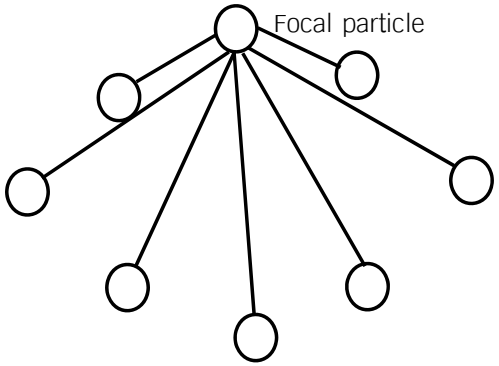


    - Ring topology



n=2

- Wheel topology



- Individual best (pbest)
  - Initialize the swarm $P(t)$ at $t=0$ of particles such that the position ($\mathbf{x}_i(t)$) of i^{th} particle ($P_i \in P(t)$) is random within the hyperspace
  - Evaluate the performance F of each particle using $\mathbf{x}_i(t)$
  - Compare the performance of each individual to its best performance
    - *If* $F(\mathbf{x}_i(t)) < pbest_i$

      $pbest_i = F(\mathbf{x}_i(t))$

      $x_{pbest_i}(t) = x_i(t)$
    - End
  - Change the velocity vector of each particle

    $v_i(t) = v_i(t-1) + \rho\left(x_{pbest_i} - x_i(t)\right)$ where ρ is positive random number
  - Move each particle to a new position

    $$x_i(t) = x_i(t-1) + v_i(t)$$
  - Set $t=t+1$
  - Repeat until converge (go to step 2)

- Global best (gbest)
  - Initialize the swarm $P(t)$ at t=0 of particles such that the position ($\mathbf{x}_i(t)$) of i^{th} particle ($P_i \in P(t)$) is random within the hyperspace
  - Evaluate the performance F of each particle using $\mathbf{x}_i(t)$
  - Compare the performance of each individual to its best performance
    - *If* $F(\mathbf{x}_i(t)) < pbest_i$

      $pbest_i = F(\mathbf{x}_i(t))$

      $\mathrm{x}_{pbest_i}(t) = \mathrm{x}_i(t)$
    - End
  - Compare the performance of each individual to its best performance
    - *If* $F(\mathbf{x}_i(t)) < gbest$

      $gbest = F(\mathbf{x}_i(t))$

      $\mathrm{x}_{gbest}(t) = \mathrm{x}_i(t)$
  - Change the velocity vector of each particle

    $$\mathrm{v}_i(t) = \mathrm{v}_i(t-1) + \rho_1\left(\mathrm{x}_{pbest_i} - \mathrm{x}_i(t)\right) + \rho_2\left(\mathrm{x}_{gbest} - \mathrm{x}_i(t)\right)$$
  - Move each particle to a new position

    $$\mathrm{x}_i(t) = \mathrm{x}_i(t-1) + \mathrm{v}_i(t)$$
  - Set t=t+1
  - Repeat until converge (go to step 2)

Where ρ_1 and ρ_2 are random number → $\rho_1 = r_1 C_1$ and $\rho_2 = r_2 C_2$ with r_1 and r_2 ~U(0,1), $C_1 + C_2 \leq 4$

- Local best (lbest)
  - Initialize the swarm $P(t)$ at t=0 of particles such that the position ($\mathbf{x}_i(t)$) of i^{th} particle ($P_i \in P(t)$) is random within the hyperspace
  - Evaluate the performance F of each particle using $\mathbf{x}_i(t)$
  - Compare the performance of each individual to its best performance
    - *If* $F(\mathbf{x}_i(t)) < pbest_i$

      $pbest_i = F(\mathbf{x}_i(t))$

      $x_{pbest_i}(t) = x_i(t)$
    - End
  - Compare the performance of each individual to its best performance
    - *If* $F(\mathbf{x}_i(t)) < lbest$

      $lbest = F(\mathbf{x}_i(t))$

      $x_{lbest}(t) = x_i(t)$
  - Change the velocity vector of each particle

    $$v_i(t) = v_i(t-1) + \rho_1\left(x_{pbest_i} - x_i(t)\right) + \rho_2\left(x_{gbest} - x_i(t)\right)$$
  - Move each particle to a new position

    $$x_i(t) = x_i(t-1) + v_i(t)$$
  - Set t=t+1
  - Repeat until converge (go to step 2)

Where ρ_1 and ρ_2 are random number → $\rho_1 = r_1 C_1$ and $\rho_2 = r_2 C_2$ with r_1 and r_2 ~U(0,1), $C_1 + C_2 \leq 4$

- Fitness calculation→can be objective function→ same idea as in GA
- Convergence→executed for a fixed number of iteration or velocity changes are close to 0 for all particles
- PSO system parameter
  - Dimension of problem
  - Number of individual
  - Upper limit of ρ
  - Upper limit on the maximum velocity (v_{max})

    If $v_{ij}(t) > v_{max}$ then $v_{ij}(t) = v_{max}$ and If $v_{ij}(t) < -v_{max}$ then $v_{ij}(t) = -v_{max}$

    where $v_{ij}(t)$ is a velocity of ith particle at jth dimension at time t

    OR if do not want to use v_{max}

$$v_i(t) = \kappa\left(v_i(t-1) + \rho_1\left(x_{pbest_i} - x_i(t)\right) + \rho_2\left(x_{gbest} - x_i(t)\right)\right)$$

Where

$$\kappa = 1 - \frac{1}{\rho} + \frac{\sqrt{\left|\rho^2 - 4\rho\right|}}{2}$$

and $\rho = \rho_1 + \rho_2 > 4.0$

- Neighborhood size
  - gbest →simply lbest with the entire swarm as the neighborhood
  - gbest →more susceptible to local minima
- Inertia weight

$$v_i(t) = \phi v_i(t-1) + \rho_1\left(x_{pbest_i} - x_i(t)\right) + \rho_2\left(x_{gbest} - x_i(t)\right)$$

Where ϕ is inertia weight ($\phi \leq 1$) and

$$\phi > \frac{1}{2}\left(c_1 + c_2\right) - 1$$

- Modified PSO
  - Using selection→ perform selection before velocity update

Algo

- for each particle in the swarm, score the performance of that particle w.r.t a random selected group of *k* particles
- Rank the particles according to these performance scores
- Select the top half of the particle and copy their current position onto that of the bottom half of the swarm without changing the personal best values of the bottom half of the swarm

- Breeding PSO

Algo

- Calculate the particle velocities and new positions
- For each particle assign a breeding probability p_b
- Select 2 particles assume P_a and P_b and produce 2 offsprings using

$$\mathbf{x}_a(t+1) = r_1 \mathbf{x}_a(t) + (1 - r_1)\mathbf{x}_b(t)$$

$$\mathbf{x}_b(t+1) = r_2 \mathbf{x}_b(t) + (1 - r_2)\mathbf{x}_a(t)$$

$$\mathbf{v}_a(t+1) = \frac{\mathbf{v}_a(t) + \mathbf{v}_b(t)}{\left\|\mathbf{v}_a(t) + \mathbf{v}_b(t)\right\|}\left\|\mathbf{v}_a(t)\right\|$$

$$\mathbf{v}_b(t+1) = \frac{\mathbf{v}_a(t) + \mathbf{v}_b(t)}{\left\|\mathbf{v}_a(t) + \mathbf{v}_b(t)\right\|}\left\|\mathbf{v}_b(t)\right\|$$

Where r_1, $r_2 \sim U(0,1)$

- Set personal best position of each particle involved in the breeding process to its current position

- Neighborhood topology
  - Use of indices
  - Use spatial neighbor

    P_b is said to be in the neighborhood of P_a if

$$\frac{\left\| \mathbf{x}_a - \mathbf{x}_b \right\|}{d_{\max}} < \xi$$

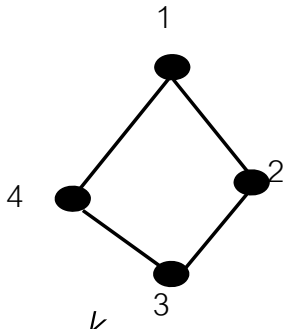
    where d_{max} is the largest distance between any 2 particles and

$$\xi = \frac{3t + 0.6t_{\max}}{t_{\max}}$$

- Ant Colony Optimization
  - tasks
    - Reproduction
    - Defense
    - Foor collection
    - Brood care
    - Nest brooming
    - Nest building
  - Natural stigmergy
    - Lack of central coordination
    - Communication and coordination among individuals in a colony are based on local modifications of the environmeny
    - Positive feedback

- Shortest path problem →traveling salesman problem (TSP)
- Virtual pheromone
- Negative feedback→avoid premature feedback
- Time scale→not too large and not too short
- Ant System (AS)
  - Find a closed tour of minimal length connected *n* given cities

$$d_{ij} = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2}$$ distance between city *i* and city *j*



Graph (N,E) With 4 cities and they are all connected

- Tabu list and J_i^k set of of all cities that ant *k* is has to visit when it is on city *i*

- AS algo

For every edge (*i*,*j*)

$\tau_{ij}(0) = \tau_0$ *#initialize pheromone*

End For

For $k = 1$ *to* m *#for each ant k*

Place ant k on a randomly chosen city

End For

Set T^+ *and* L^+

For $t = 1$ *to* t_{max}

For $k = 1$ *to* m

build tour $T^k(t)$ *by applying* $n-1$ *times the following steps: choose the next city j* when the current city is city *i* using the transition rule

$$p_{ij}^k(t) = \begin{cases} \dfrac{\left[\tau_{ij}(t)\right]^\alpha \left[\eta_{ij}\right]^\beta}{\sum_{l \in J_i^k} \left[\tau_{il}(t)\right]^\alpha \left[\eta_{il}\right]^\beta} & \text{if } j \in J_i^k \\ 0 & \text{else} \end{cases}$$

Where α *and* β *are adjustable and visibility* $\eta_{ij} = \dfrac{1}{d_{ij}}$

End For

For $k = 1$ *to* m

compute length $L^k(t)$ *of tour* $T^k(t)$ *produced by ant k*

End For

If an improved tour is found then

update T^+ *and* L^+

End If

For every edge (*i,j*)

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) + e\Delta\tau^{e}_{ij}(t)$$

where

$$\Delta\tau^{e}_{ij}(t) = \begin{cases} \dfrac{Q}{L^{+}} & \text{if } (i,j) \in T^{+} \\ 0 & \text{if } (i,j) \notin T^{+} \end{cases}$$

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau^{k}_{ij}(t)$$

$$\Delta\tau^{k}_{ij}(t) = \begin{cases} \dfrac{Q}{L^{k}(t)} & \text{if } (i,j) \in T^{k}(t) \\ 0 & \text{if } (i,j) \notin T^{k}(t) \end{cases}$$

End For

End For

- Ant colony system (ACS)
  - Different transition rule
  - Different pheromone trail update rule
  - Use of local updates of pheromone trail to favor exploration
  - Use of candidate list→ cl closest cities ordered by increasing distance→ ant first restricts the choice of the next city to those in the list, and consider other cities if all in the list have been visited

For every edge (*i*,*j*)

$\tau_{ij}(0) = \tau_0$ *#initialize pheromone*

End For

For k= 1 *to m #for each ant k*

Place ant k on a randomly chosen city

End For

Set T^+ *and* L^+

For t= 1 *to* t_{max}

For k= 1*to m*

build tour $T^k(t)$ *by applying* $n-1$ *times the following steps:*

if exits at least one city j∈*candidate list then choose the next cityt* $j \in J_i^k$ *among cl citiesy in the candicate list as*

$$j = \begin{cases} \arg\max_{u \in J_i^k} \left\{ [\tau_{iu}(t)][\eta_{iu}]^\beta \right\} & \text{if } q \le q_0 \\ J & \text{if } q > q_0 \end{cases}$$

where $q \sim U(0,1)$, $0 \le q_0 \le 1$

where $J \in J_i^k$ is chosen according to $p_{iJ}^k(t) = \dfrac{[\tau_{iJ}(t)][\eta_{iJ}]^\beta}{\sum_{l \in J_i^k} [\tau_{il}(t)][\eta_{il}]^\beta}$

else choose the closest city $j \in J_i^k$

end if

after each transition ant k *applies the local update rule*

$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t) + \rho\tau_0$ *where* $\tau_0 = (nL_{nn})^{-1}$

End For

For $k = 1$ *to* m

compute length $L^k(t)$ *of tour* $T^k(t)$ *produced by ant* k

End For

If an improved tour is found then

update T^+ *and* L^+

End If

For every edge (i,j)

$$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}(t)$$

where

$$\Delta\tau_{ij}(t) = \frac{1}{L^+}$$

End For

End For