Lecture 3 - Selection Schemes – *Parent Selection and Survivor Selection*

Yuhui Shi

CSE, SUSTech

Outline of This Lecture

- Recap of The Last Lecture
- Parent Selection Schemes
  - Parent Selection: Basic Schemes
  - Parent Selection: Other Schemes
- Survivor Selection
- Example: A Hybrid EA With Local Search
- Summary of this Lecture

Outline of This Lecture

- **Recap of The Last Lecture**
- Parent Selection Schemes
  - Parent Selection: Basic Schemes
  - Parent Selection: Other Schemes
- Survivor Selection
- Example: A Hybrid EA With Local Search
- Summary of this Lecture

Recall: Components of EA

- Representation: How to represent a solution using a chromosome/genome?
- Population: Population-based algorithm.
- Evaluation function: Evaluate the quality, the fitter the better.
- Initialisation: Randomly or based on a priori knowledge.
- Parent selection mechanism: Choose better individuals for reproduction.
- Variation operators: Recombination/Crossover and mutation.

  → Stochastic algorithm.
- Survivor selection mechanism: Survival of the fittest!
- Termination: When to stop.

Recall: Main Steps of Evolutionary Algorithms

1. Initialise the population at random
2. REPEAT
   a. Evaluate fitness of individuals in the population
      (μ is the population size)
   b. Compute the selection probability for each individual
   c. REPEAT
      I. Select two individuals as parents according to the probabilities in Step b
      II. Crossover the two individuals in Step 2.c.I with a crossover rate
          /* After this step, we have two individuals */
      III. Mutate the two individuals in Step 2.c.II with a mutation rate
      UNTIL we have obtained μ new individuals
   d. Use the μ new individuals to replace the previous population

UNTIL stopping criteria are met

/* Output the best individual in the population */

Recall: Previous Lecture

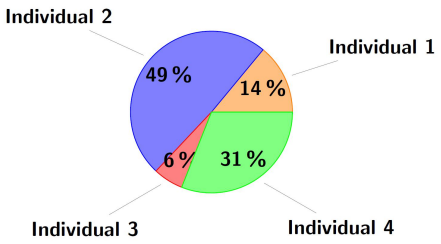
- Crossover/recombination operators and mutation operators for
  - ✓ discrete representation; and
  - ✓ real-valued representation.

→ Mutation is the only way to introduce new gene values.

This Lecture: Selection Schemes

Important Notion: Selection Pressure

- Selection pressure: "... as selection pressure increases, so fitter solutions are more likely to survive, or be chosen as parents, and less-fit solutions are correspondingly less likely." [1]
- A selection scheme determines the probability distribution for selecting individuals, aiming at changing the selection pressure.
  - ✓Uniform probability distribution → weak selection pressure.
  - ✓Probabilities biased towards better individuals
    - → higher selection pressure
    - → the better individuals are preferred more.



[Exercise]

Read Section 5.4 (pages 90-91) of [1] for more about Selection Pressure.

Selection Categories

- Parent Selection:
  - ✓Before search operators, from the current population of μ individuals, select λ individuals to reproduce/generate the next generation.
  - ✓$\lambda < \mu$
  - ✓The key issue here is to generate a probability distribution so that individuals can be selected according to such a distribution.
- Survivor Selection:
  - ✓After search operators, from the μ parents + λ generated offspring, select μ individuals to form the next generation.
  - ✓This is also called the (μ+λ) selection strategy.

Outline of This Lecture

- Recap of The Last Lecture
- **Parent Selection Schemes**
  - Parent Selection: Basic Schemes
  - Parent Selection: Other Schemes
- Survivor Selection
- Example: A Hybrid EA With Local Search
- Summary of this Lecture

Parent Selection: Basic Schemes

- Fitness proportional selection (FPS) schemes
  - ✓ Roulette wheel selection (fitness proportional selection)
  - ✓ FPS with scaling
    - ➢ simple scaling
    - ➢ sliding window
    - ➢ sigma scaling
    - ➢ power scaling
    - ➢ exponential scaling, etc.
- Rank-based selection schemes
  - ✓ Simple rank-based selection
  - ✓ Linear ranking selection
  - ✓ Exponential ranking selection
  - ✓ Power ranking
  - ✓ Geometric ranking, etc.
- Tournament selection

Remarks and Notations

- P: Current population.
- μ: Population size, i.e., $\mu = |P|$.
- x: An individual, a vector.
- x_i: The i^{th} coordinate of the vector x.
- f(x): Fitness value of the individual x.
- Prob(x): Probability of selecting the individual x.
- We always maximise the fitness values f(·) in this section.
- When ranking a population, the individuals are ranked from worst to best by their fitness values.
- rank(x): Ascending rank of the individual x in the population, always ranges from 0 to $\mu - 1$ in this section.

$f(x^{(\mu-1)}) \geq f(x^{(\mu-2)}) \cdots \geq f(x^{(0)})$, where $x^{(i)}$ is the individual with rank i.

Larger rank → better individual!

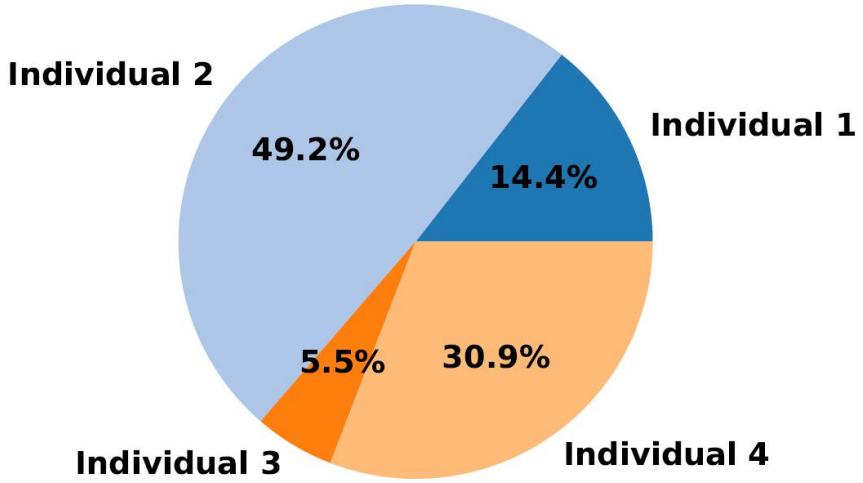
Roulette Wheel Selection

- Also known as "Fitness Proportional Selection".
- Probability of selecting the individual x:

$$Prob(\mathbf{x}) = \frac{f(\mathbf{x})}{\sum_{\mathbf{x}' \in \mathcal{P}} f(\mathbf{x}')}.$$

- Example: We maximise f!

Idx	x	fitness	$Prob(\mathbf{x})$
1	13	169	0.144
2	24	576	0.492
3	8	64	0.055
4	19	361	0.309



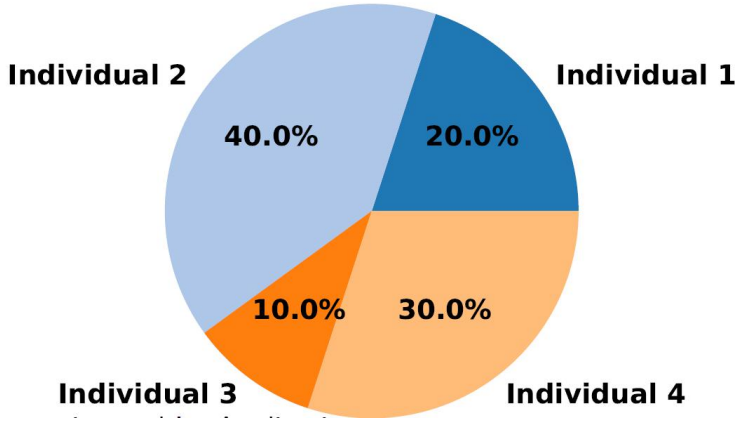
Rank-based Selection

- Rank the individuals from worst to best.
  - ✓ $\forall x \in P$, rank(x) $\in \{0, \ldots, \mu - 1\}$.
  - ✓ Larger rank refers to better individual!
- Probability of selecting the individual x:

$$Prob(\mathbf{x}) = \frac{rank(\mathbf{x}) + 1}{\sum_{\mathbf{x}' \in \mathcal{P}} (rank(\mathbf{x}') + 1)}.$$

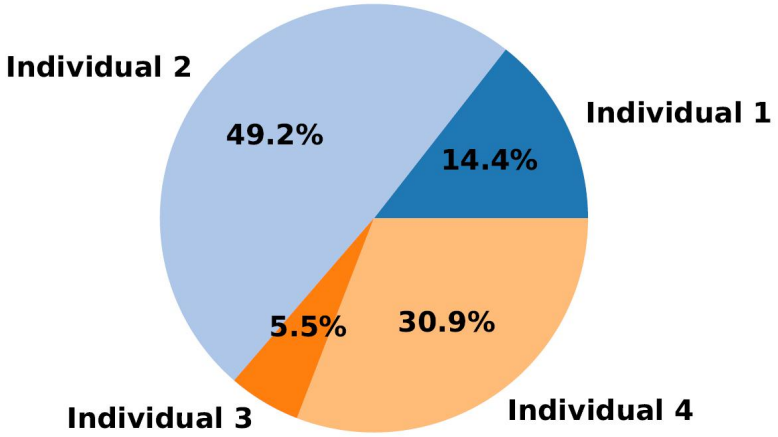
- Example: We maximise f!

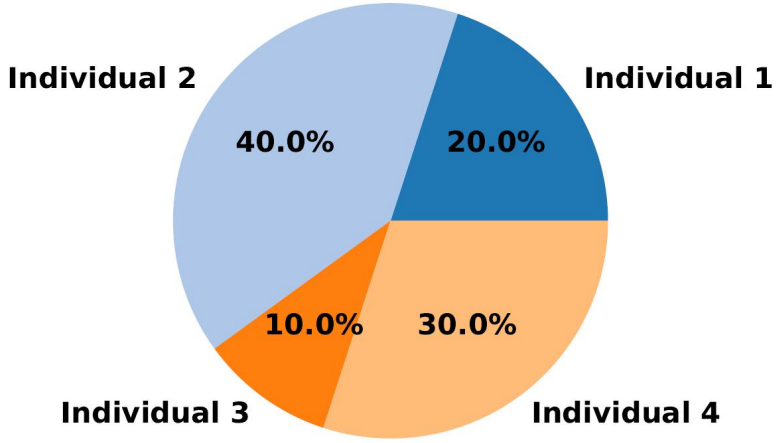
Idx	x	fitness	rank	$Prob(\mathbf{x})$
1	13	169	1	0.2
2	24	576	3	0.4
3	8	64	0	0.1
4	19	361	2	0.3



Roulette Wheel (left) vs. Rank-based (right)

Idx	x	fitness	rank
1	13	169	1
2	24	576	3
3	8	64	0
4	19	361	2





Tournament Selection

1. Select k individuals from the population P at random.
2. Select the best individual (the one with the highest fitness) from the k individuals selected above.

- k: tournament size.
- Increasing k → increased selection pressure.

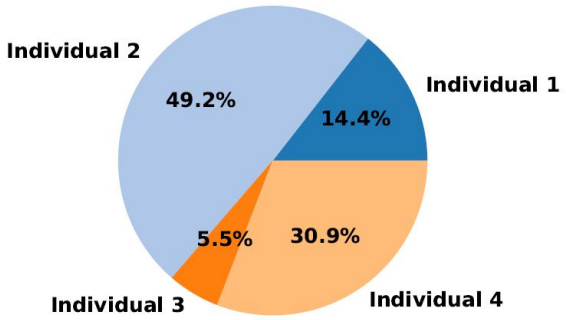
Roulette Wheel Selection: Revisited

- Recap: Probability of selecting the individual x:

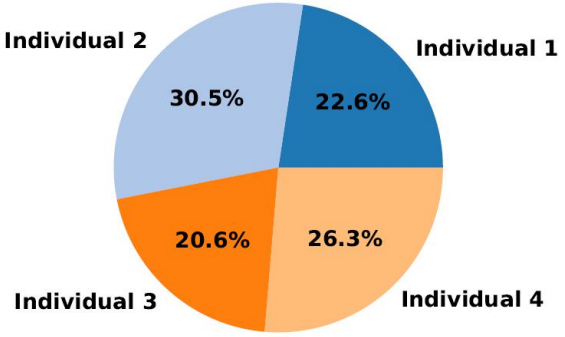
$$Prob(\mathbf{x}) = \frac{f(\mathbf{x})}{\sum_{\mathbf{x}' \in \mathcal{P}} f(\mathbf{x}')}.$$

- What are the (potential) issues of Roulette Wheel Selection? E.g., what if the fitness values are close to each other?
- Example: We maximise f!

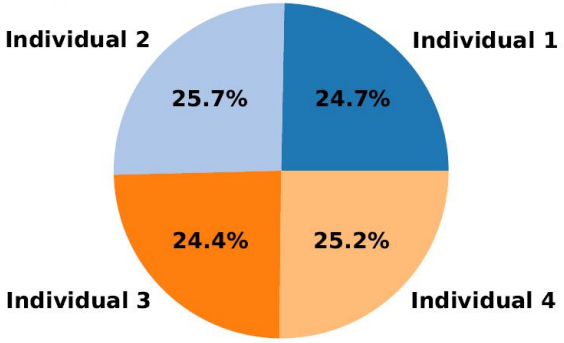
$f(\mathbf{x}) = \mathbf{x}^2$



$f(\mathbf{x}) = \mathbf{x}^2+1000$



$f(\mathbf{x}) = \mathbf{x}^2+10000$



Roulette Wheel Selection: Revisited
-(Potential) Issues

1. Does not allow negative fitness values.
2. Premature convergence (过早收敛): at least one individual is outstanding, thus, its fitness is much better than others'.
   → Domination of super individuals in early generations and slow convergence in later generations.
3. No selection pressure: the fitness values are close to each other.
4. The selection behaves differently if the fitness function is transposed.

- Fitness scaling has often been used in early days to combat the 1st and 2nd issues.

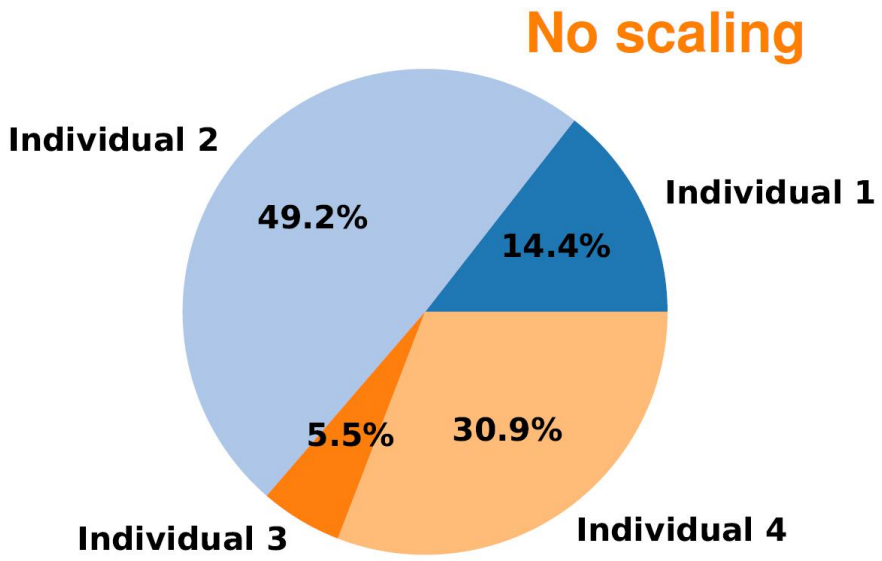
Fitness Scaling: Simple Scaling

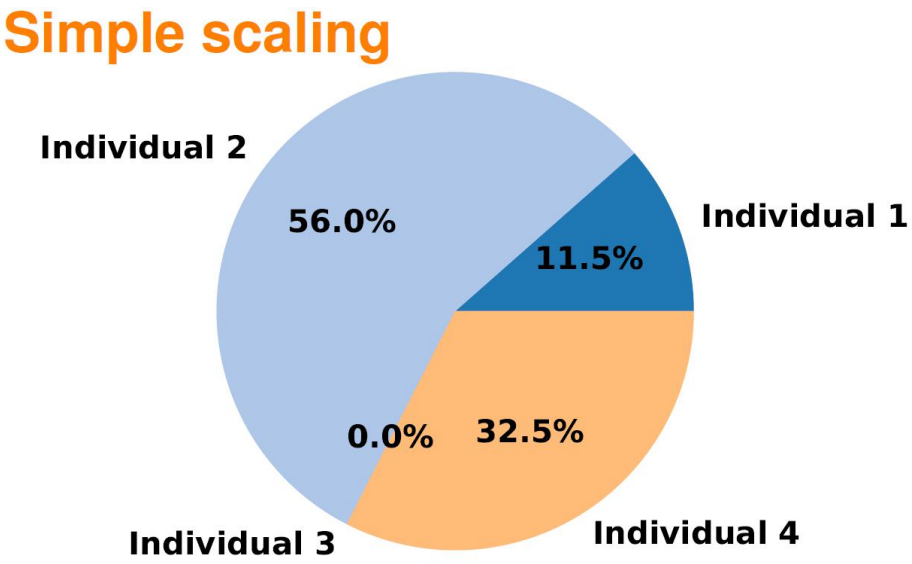
- The new fitness of an individual x is

$$f_{scaled}(x) = f(x) - f_{worst}$$

where f_{worst}, is the fitness of the worst individual at the current generation.

- Example: We maximise f!





Fitness Scaling: Sliding Window

- Considering a window width of *w*, the sliding windowing fitness of an individual x of the population at generation *t* is

$$f_{scaled}(x) = f(x) - \eta,$$

where

- ✓ P_i denotes the population at generation i,
- ✓ $\eta = \min_{i \in \{t-w+1,\ldots,t\}} \min_{x' \in p_i} f(x').$

→ Fitness of the worst individual during the latest *w* generations!

- Special case:

Let $w = 0$, then $f_{scaled}(x) = f(x) - \eta$, where $\eta = \min_{x' \in p_t} f(x')$

⇒ Simple Fitness Scaling

Fitness Scaling: Sigma Scaling

- The new fitness of an individual x is

$$f_{scaled}(x) = \max\{f(x) - (\bar{f} - c.\sigma_f), 0\},$$

where

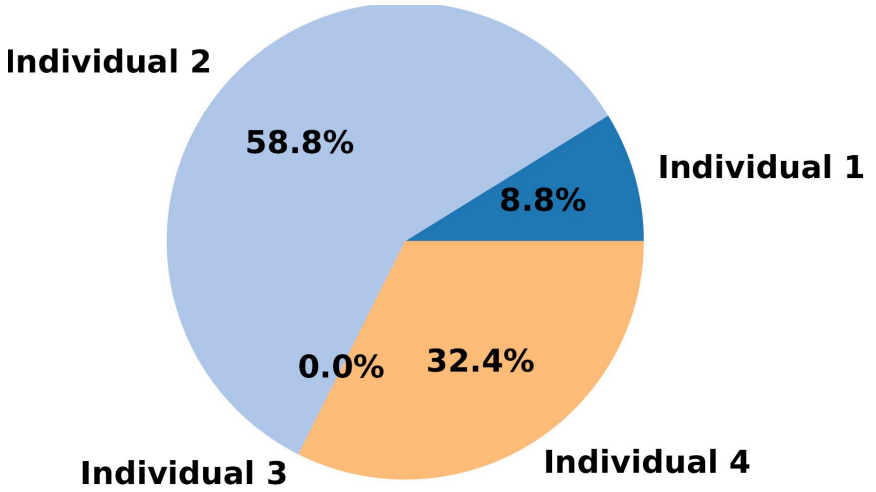
✓ $\bar{f} = \frac{1}{\mu}\sum_{x' \in P} f(x')$, thus the average fitness in current population,

✓ σ_f is the standard deviation of fitness in current population P,

✓ c is a constant,

✓ μ is the population size.

- Example with c = 1: We maximise f!



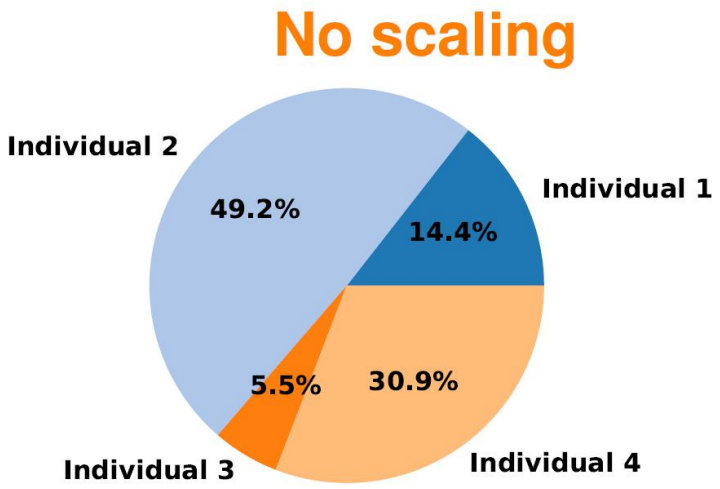
Fitness Scaling: Power Scaling

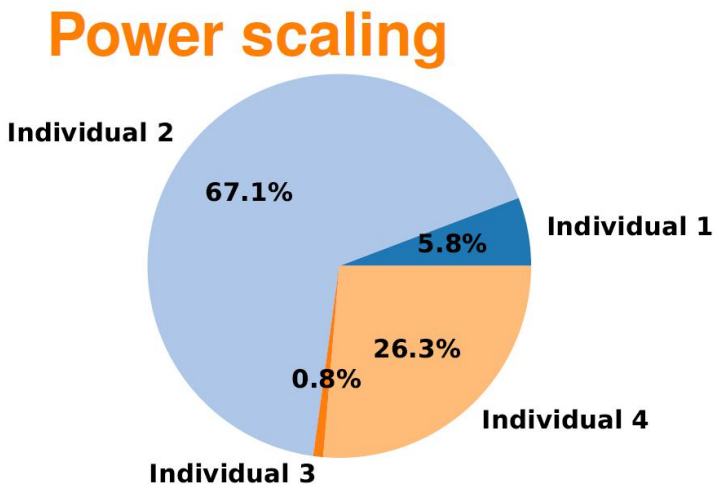
- The new fitness of an individual x is

$$f_{scaled}(x) = (f(x))^k,$$

  usually k > 1.

- Example with k = 2: We maximise f!





Fitness Scaling: Exponential Scaling

- The new fitness of an individual x is

$$f_{scaled}(x) = \exp(\frac{f(x)}{T}),$$

where T > 0 is a constant.

Recall: Simple Rank-based Selection

- Assume that
  - ✓the population size is μ,
  - ✓rank 0 indicates the worst individual,
  - ✓rank μ − 1 indicates the best individual.
- Probability of selecting the individual x:

$$Prob(\mathbf{x}) = \frac{rank(\mathbf{x}) + 1}{\sum_{\mathbf{x}' \in \mathcal{P}} (rank(\mathbf{x}') + 1)}.$$

- Remark: there are many ways of mapping rank number to selection probability.

Alternative 1: Linear Ranking Selection

- The probability of selecting the individual x is:

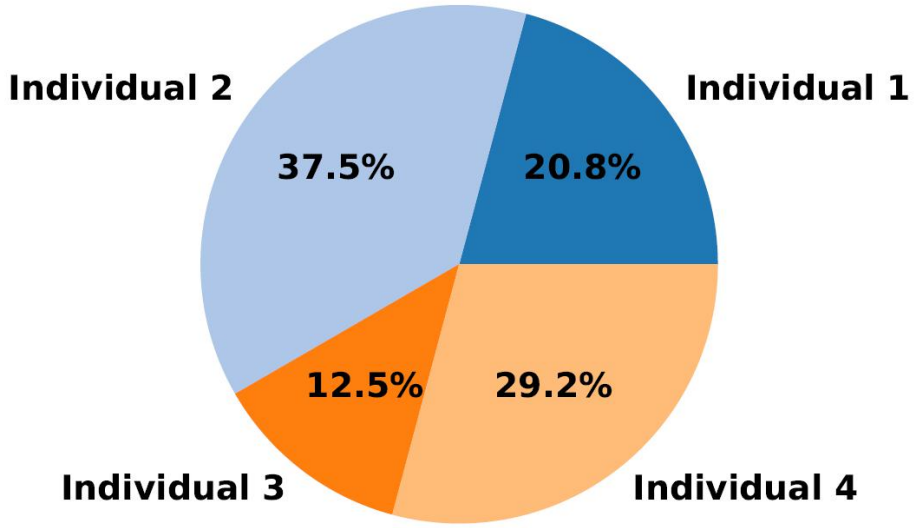
$$Prob(\mathbf{x}) = \frac{\alpha + (\beta - \alpha)\frac{rank(\mathbf{x})}{\mu - 1}}{\mu}.$$

  - ✓ *rank(x)* $\in \{0, \ldots, \mu - 1\}$, μ is the population size ($\mu = |P|$);
  - ✓ α (β) indicates the expected number of offspring produced by the worst (best) individuals.
  - ✓ Note that $\sum_{x \in P} Prob(x) = 1$, hence, α + β = 2. So 1 ≤ β ≤ 2 and α = 2 − β.
- Remark: only limited selection pressure can be applied.
  - → Use Exponential Ranking Selection if higher selection pressure is required.

Alternative 1: Linear Ranking Selection
-Example with α = 0.5, β = 1.5

Recall: $$Prob(\mathbf{x}) = \frac{\alpha + (\beta - \alpha)\frac{rank(\mathbf{x})}{\mu - 1}}{\mu}.$$ (We maximise f!)

Idx	x	fitness	rank
1	13	169	1
2	24	576	3
3	8	64	0
4	19	361	2

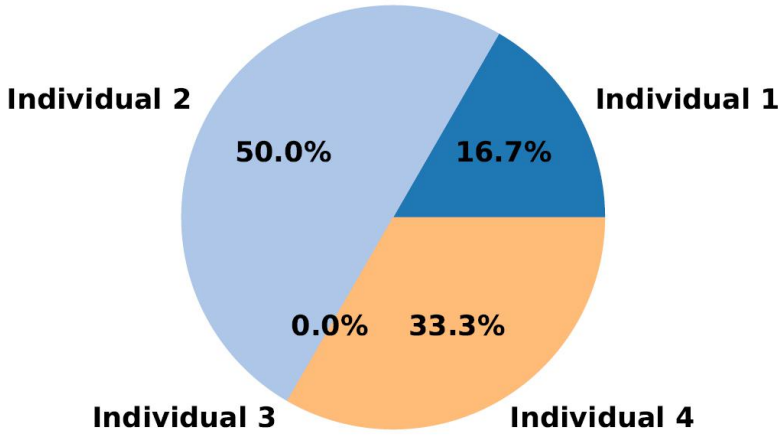


Alternative 1: Linear Ranking Selection
-Comparison with Different Parameters

- $\alpha = 0, \beta = 2$

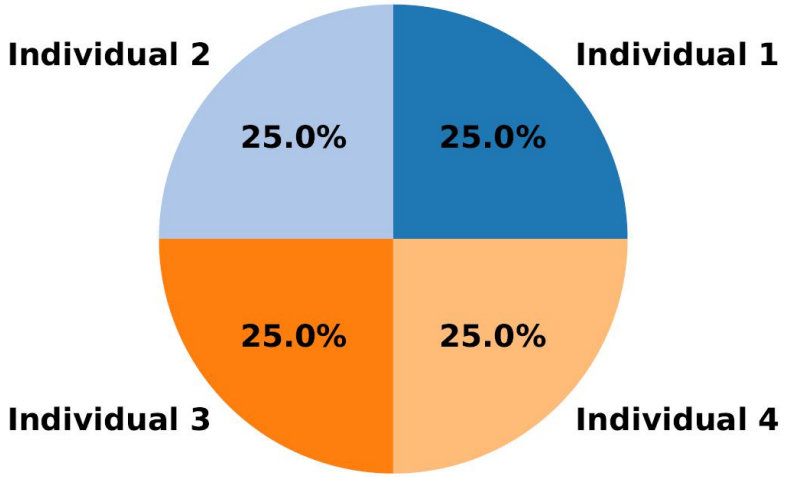
  → Simple rank-based selection

  (*with modification*)



- $\alpha = 1, \beta = 1$

  → Uniform selection



Alternative 2: Exponential Ranking Selection

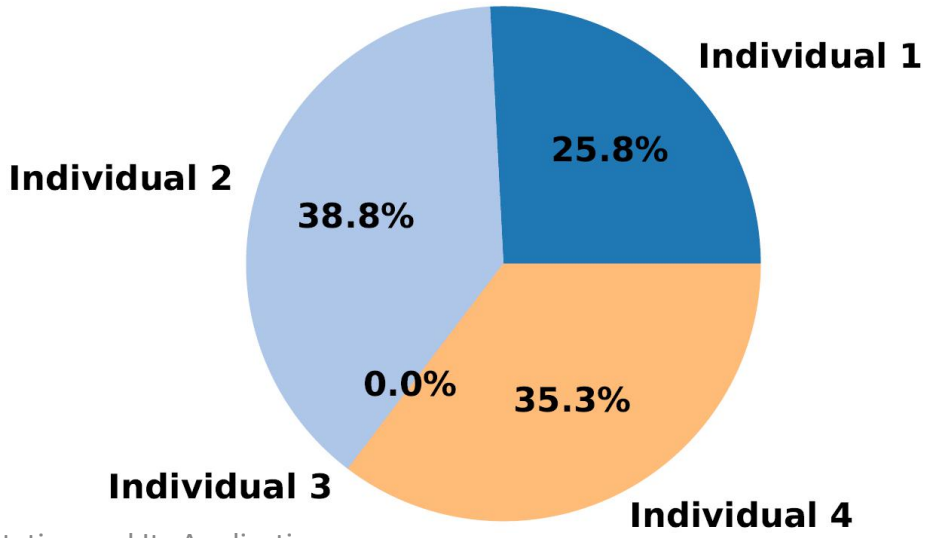
- The probability of selecting the individual x is:

$$Prob(\mathbf{x}) = \frac{1 - e^{-rank(\mathbf{x})}}{C}$$

where C is a normalising factor which ensures that the sum of the probabilities is 1. $C = \mu - \frac{e - e^{-\mu+1}}{e-1}$ i.e., a function of the population size.

- Example:

Idx	x	fitness	rank
1	13	169	1
2	24	576	3
3	8	64	0
4	19	361	2



Alternative 3: Power Ranking

- The probability of selecting the individual x is:

$$Prob(\mathbf{x}) = \frac{\alpha + (\beta - \alpha) \cdot (\frac{rank(\mathbf{x})}{\mu - 1})^k}{C},$$

where

✓μ is the population size,

✓k > 0, α, β are parameters, 1 ≤ β ≤ 2 and α = 2 − β,

✓C is a normalising factor which ensures that the sum of the probabilities is 1.

- When k = 1 and C = μ → Linear Ranking Selection

$$Prob(\mathbf{x}) = \frac{\alpha + (\beta - \alpha) \frac{rank(\mathbf{x})}{\mu - 1}}{\mu}.$$

Alternative 4: Geometric Ranking

- The probability of selecting the individual x is:

$$Prob(\mathbf{x}) = \frac{\alpha(1-\alpha)^{\mu-1-rank(\mathbf{x})}}{C}.$$

where

✓μ is the population size,

✓α is a parameter ($0 < \alpha < 1$),

✓C is a normalising factor which ensures that the sum of the probabilities is 1. Thus, $C = 1 - (1 - \alpha)^{\mu}$.

Outline of This Lecture

- Recap of The Last Lecture
- Parent Selection Schemes
  - Parent Selection: Basic Schemes
  - Parent Selection: Other Schemes
- **Survivor Selection**
- Example: A Hybrid EA With Local Search
- Summary of this Lecture

Recall: Main Steps of Evolutionary Algorithms

1. Initialise the population at random
2. REPEAT
   a. Evaluate fitness of individuals in the population (μ is the population size)
   b. Compute the selection probability for each individual
   c. REPEAT
      I. Select two individuals as parents according to the probabilities in Step b
      II. Crossover the two individuals in Step 2.c.I with a crossover rate
      /* After this step, we have two individuals */
      III. Mutate the two individuals in Step 2.c.II with a mutation rate
      UNTIL we have obtained μ new individuals
   d. Use the μ new individuals to replace the previous population

UNTIL stopping criteria are met

/* Output the best individual in the population */

Survivor Selection

- All the presented parent selection schemes can be applied for selecting survivors.
- However, there are some particular survivor selection schemes, in which replacement is explicit.

  → For this reason, the schemes are often referred to as "replacement strategies".

  ✓Age-based

  ✓Fitness-based

Age-based Replacement

- Core ideas:
  - ✓Each individual survives for the same number of generations.
  - ✓Fitness is not taken into account.

  → Therefore, the best fitness of any given generation could be worse than that of its predecessor.
  - ✓A net increase in the $\bar{f}$ over time relies on
    1. having sufficient selection pressure;
    2. using variation operators that are not too disruptive.
- Can be used in all EAs.
  - ✓$\mu = \lambda$: Generational model.
    - ➢ Offspring replace the whole population.
  - ✓$\lambda < \mu$.
    - ➢ Example: Steady-State GAs
    - ➢ Delete-oldest (FIFO) strategy

Fitness-based Replacement Strategies

- Replace worst
- Elitism
- Round-robin tournament
- (μ + λ) selection
- (μ,λ) selection

Replace Worst

- The worst λ individuals of the population are selected for replacement.
- Potential issue: premature convergence.
- Solutions: use with
  - ✓large populations, and/or
  - ✓“no duplicates” policy: replace the most similar

Elitism

- Always copy the best individuals from one generation to the next without going through any evolutionary operators.

Round-robin Tournament

- Introduced within Evolutionary Programming (EP).
- Used for
  - ✓choosing μ survivors, or
  - ✓choosing λ parents from a given population of μ.
- Pairwise tournament competitions in round-robin format.
  - ✓Each individual is evaluated against m others randomly chosen from the merged parent and offspring populations ($\mu + \lambda$).
  - ✓For each comparison, a “win” is assigned if the individual is better than its opponent.
  - ✓Finally, the μ individuals with the greatest number of wins are selected.
- m controls the selection pressure.

$(\mu + \lambda)$ Selection vs. (μ,λ) Selection

- Both come from Evolution Strategies (ES).
- Both for survivor selection
- λ offspring are generated using the μ parents.
- Differences:

$(\mu + \lambda)$ Selection:

- $(\mu + \lambda)$ individuals ranked by fitness, the top μ to form the next generation.

(μ,λ) Selection:

- Mixture of age and fitness.
  - ✓ Age: all parents are discarded.
  - ✓ Fitness: λ Offspring ranked by fitness, the top μ form the next generation.

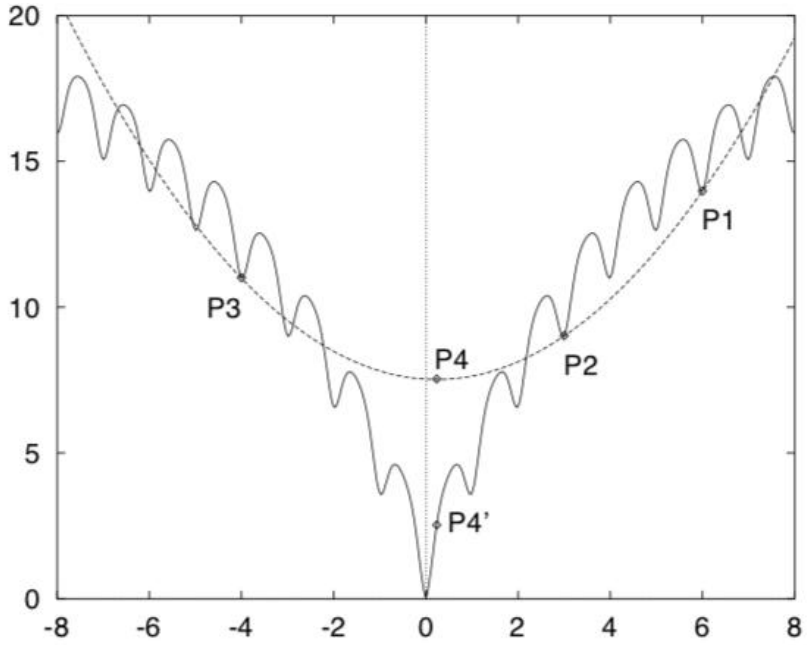
Outline of This Lecture

- Recap of The Last Lecture
- Parent Selection Schemes
  - Parent Selection: Basic Schemes
  - Parent Selection: Other Schemes
- Survivor Selection
- **Example: A Hybrid EA With Local Search**
- Summary of this Lecture

What Does Quadratic Recombination Mean?

$$x_{4,j} = \frac{1}{2} \cdot \frac{(x_{2,j}^2 - x_{3,j}^2) f(\mathbf{x}_1) + (x_{3,j}^2 - x_{1,j}^2) f(\mathbf{x}_2) + (x_{1,j}^2 - x_{2,j}^2) f(\mathbf{x}_3)}{(x_{2,j} - x_{3,j}) f(\mathbf{x}_1) + (x_{3,j} - x_{1,j}) f(\mathbf{x}_2) + (x_{1,j} - x_{2,j}) f(\mathbf{x}_3)}.$$

- Note that we are minimising "fitness" here.



A Hybrid EA With Local Search

1. Initialise μ individuals uniformly at random.
2. Perform local search from each individual.
3. REPEAT
   I. Generate 3 points P_1, P_2, P_3 by global discrete recombination.
   II. Perform a quadratic approximation using P_1, P_2 , P_3 to produce a point P_4.
   III. Perform a local search from P_4 and update P_4 with the search result. (Sounds Lamarkian)
   IV. Place P_1, P_2, P_3, P_4 into the population and do a ($\mu + 4$) truncation selection.
4. UNTIL termination criteria are met.

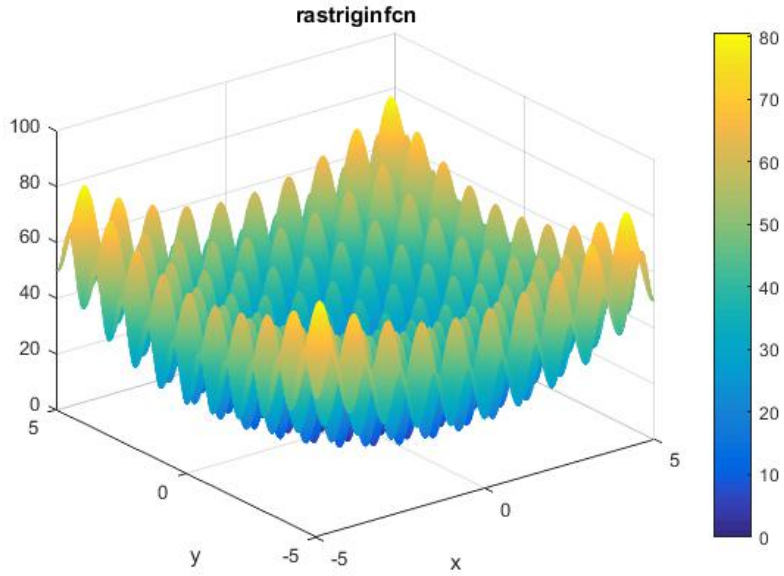
Local Search With Random Memorising

- Store best solutions in a memory.
- Retrieve a random one (old best) when a new best solution is found.
- Search along the direction of old−→new, i.e., the direction of

$$s := \frac{\mathbf{x}_{new} - \mathbf{x}_{old}}{\left\|\mathbf{x}_{new} - \mathbf{x}_{old}\right\|}$$

Experimental Studies

- 18 multimodal benchmark functions.
- Population size 30.
- Maximum function evaluation 500, 000.
- 50 independent runs for each function.



(Figure from http://benchmarkfcns.xyz/benchmarkfcns/)

18 Benchmark Functions (f_8 – f_{15})

Test function	n	S	f_{min}
$f_8(\mathbf{x}) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	**30**	$[-500, 500]^n$	**-12569.5**
$f_9(\mathbf{x}) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10)]$	**30**	$[-5.12, 5.12]^n$	**0**
$f_{10}(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i\right) + 20 + e$	**30**	$[-32, 32]^n$	**0**
$f_{11}(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	**30**	$[-600, 600]^n$	**0**
$f_{12}(\mathbf{x}) = \frac{\pi}{n}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$, $y_i = 1 + \frac{1}{4}(x_i + 1)$, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \le x_i \le a, \\ k(-x_i - a)^m, & x_i < -a. \end{cases}$	**30**	$[-50, 50]^n$	**0**
$f_{13}(\mathbf{x}) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	**30**	$[-50, 50]^n$	**0**
$f_{14}(\mathbf{x}) = \left[\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right]^{-1}$	**2**	$[-65.536, 65.536]^n$	**1**
$f_{15}(\mathbf{x}) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	**4**	$[-5, 5]^n$	**0.0003075**

18 Benchmark Functions ($f_{16} - f_{25}$)

Test function	n	S	f_{min}
$f_{16}(\mathbf{x}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	**2**	$[-5,5]^n$	**-1.0316285**
$f_{17}(\mathbf{x}) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	**2**	$[-5,10] \times [0,15]$	**0.398**
$f_{18}(\mathbf{x}) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	**2**	$[-2,2]^n$	**3**
$f_{19}(\mathbf{x}) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{4} a_{ij}(x_j - p_{ij})^2\right]$	**4**	$[0,1]^n$	**-3.86**
$f_{20}(\mathbf{x}) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right]$	**6**	$[0,1]^n$	**-3.32**
$f_{21}(\mathbf{x}) = -\sum_{i=1}^{5}[(x - a_i)^T(\mathbf{x} - a_i) + c_i]^{-1}$	**4**	$[0,10]^n$	$-1/c_1$
$f_{22}(\mathbf{x}) = -\sum_{i=1}^{7}[(x - a_i)^T(\mathbf{x} - a_i) + c_i]^{-1}$	**4**	$[0,10]^n$	$-1/c_1$
$f_{23}(\mathbf{x}) = -\sum_{i=1}^{10}[(x - a_i)^T(\mathbf{x} - a_i) + c_i]^{-1}$ **where** $c_1 = 0.1$	**4**	$[0,10]^n$	$-1/c_1$
$f_{24}(\mathbf{x}) = 0.5 + \frac{\sin^2\sqrt{x_1^2+x_2^2} - 0.5}{[1.0+0.001(x_1^2+x_2^2)]^2}$	**2**	$[-100,100]^n$	0
$f_{25}(\mathbf{x}) = (x_1^2 + x_2^2)^{0.25}[\sin^2(50(x_1^2 + x_2^2)^{0.1}) + 1.0]$	**2**	$[-100,100]^n$	0

Results on f_8 –f_{13}

Prob	Eval.	Mean	StdDev	Global hit
8	199244	-11834.65	298.78	1/50
9	306280	2.67	1.58	5/50
10	370686	2.08e-12	2.11e-12	50/50
11	173592	1.87e-10	1.31e-9	50/50
12	476245	5.84e-9	2.16e-8	50/50
13	504212	1.19e-2	3.01e-2	33/50

Results on $f_8 - f_{13}$ With Population Size 50 and 60

Prob.	Pop. size	Eval.	Mean	Std Dev	Global hit
8	50	327434	-12296.68	119.27	1/50
8	60	391634	-12358.64	166.33	12/50
9	50	508021	2.98e-1	4.97e-1	36/50
9	60	610163	2.19e-1	4.12e-1	39/50

Results on $f_{14} - f_{25}$

Prob	Eval.	Mean	StdDev	Global hit
14	3052	1.04	0.20	48/50
15	111748	3.0749e-4	2.45e-10	50/50
16	2817	-1.031628	2.70e-8	50/50
17	5496	0.3979	8.26e-9	50/50
18	4676	3	0	50/50
19	6852	-3.86	0	50/50
20	17504	-3.32	2.35e-7	50/50
21	13790	-10.1532	0	50/50
22	13354	-10.4029	2.16e-7	50/50
23	14312	-10.5364	4.58e-7	50/50
24	10754	3.89e-4	1.90e-3	48/50
25	15614	1.07e-4	7.51e-4	50/50

Take Home Messages So Far

- Different problems require different search operators and selection schemes. There is no universally best one.
- Using real-valued vectors is usually more appropriate than binary strings for function optimisation.
- Many search operators are heuristics-based. Domain knowledge can often be incorporated into search operators and representation.

Outline of This Lecture

- Recap of The Last Lecture
- Parent Selection Schemes
  - Parent Selection: Basic Schemes
  - Parent Selection: Other Schemes
- Survivor Selection
- Example: A Hybrid EA With Local Search
- **Summary of this Lecture**

Selection and Reproduction: Summary

- Selection is not normally regarded as a search operator although it does influence search significantly.
- Selection can be either before or after search operators.
- When selection is used after search operators, the process of choosing the next generation from the union of all parents and offspring is sometimes called survivor selection or reproduction.
- In terms of reproduction strategies, the two extremes are generational EAs and steady-state EAs.
- The generational gap of an EA refers to the non-overlap (i.e., individuals that go through search operators) between the old and new generations.
- Elitism copies the best individuals to the next generation.

Essential Reading for This Lecture

1. T. B¨ack, D. B. Fogel, and Z. Michalewicz (eds.), Handbook of Evolutionary Computation, IOP Publ. Co. & Oxford University Press, 1997. Part C2.1-C2.6.

   (The part of selection schemes.)

2. K.-H. Liang, X. Yao and C. Newton, "Evolutionary search of approximated N-dimensional landscapes," International Journal of Knowledge-Based Intelligent Engineering Systems, 4(3):172-183, July 2000.

   https://www.cs.bham.ac.uk/~xin/papers/esna.pdf

Other Reference

1. A. E. Eiben and J. E. Smith. Introduction to evolutionary computing. Vol. 53. Springer, 2003. (Pages 31-33, pages 80-91)

Essential Reading for Next Lecture

1. X. Yao, Y. Liu and G. Lin, "Evolutionary programming made faster," IEEE Transactions on Evolutionary Computation, 3(2):82-102, July 1999.
2. C. Y. Lee and X. Yao, "Evolutionary programming using the mutations based on the L´evy probability distribution," IEEE Transactions on Evolutionary computation, vol. 8, no. 1, pp. 1–13, 2004.