9 Fitness Selection and Population Management

Fitness Selection and Population Management

- in this set of slides we'll look at:
  - population management models
  - selection operators
  - preserving diversity

Recap: General Scheme of an EA

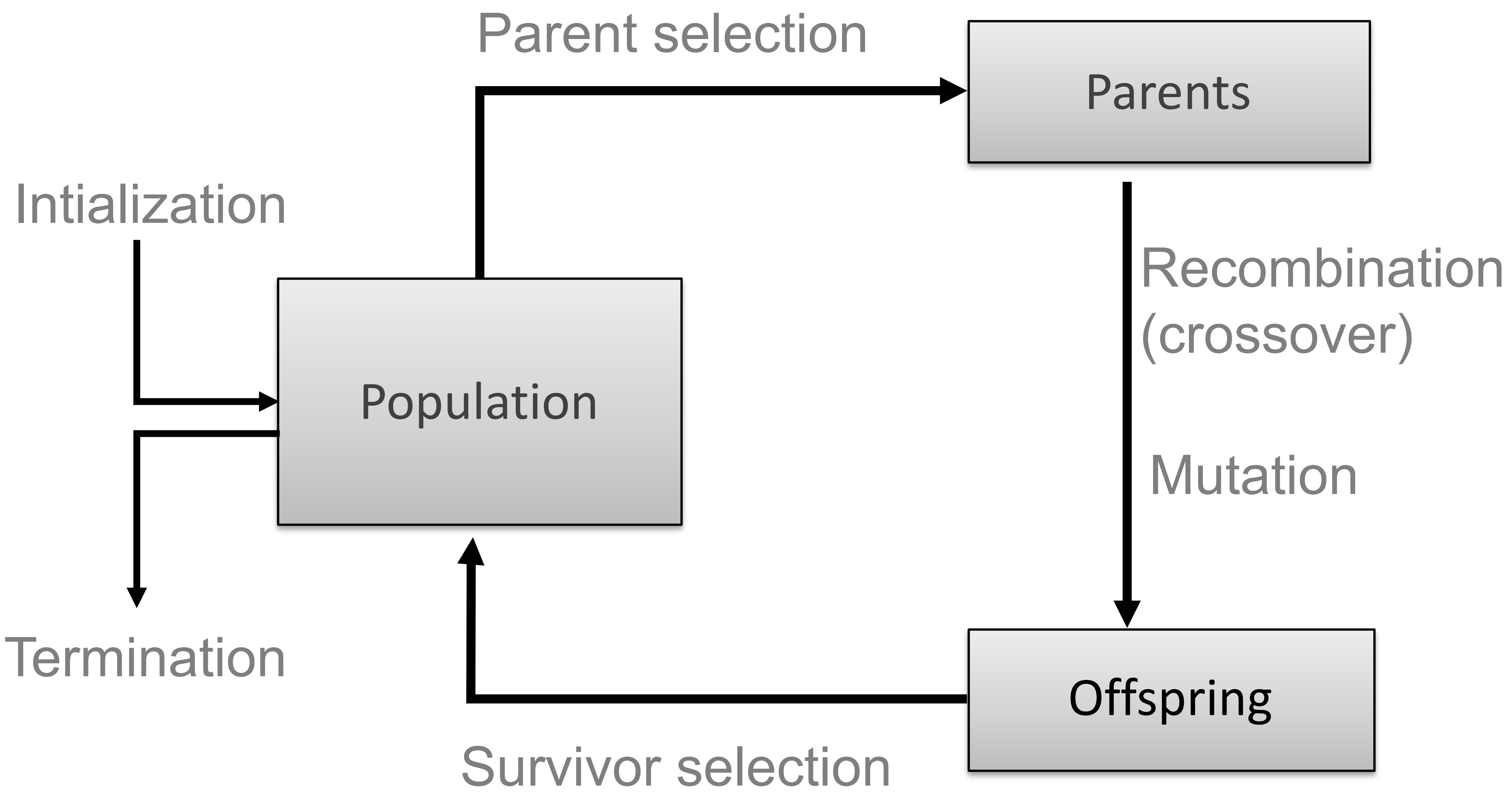


figure 3.2, Introduction to Evolutionary Computation

Population Management Models: Introduction

- two different population management models exist:

  generational model

  - each individual survives for exactly one generation
  - the entire set of parents is replaced by the offspring

  steady-state model

  - only some members of the population survive between generations
  - population has fixed size μ
  - λ new offspring are generated every generation

  generation gap

  - the proportion of the population replaced

Population Management Models: Fitness Based Competition

- selection can occur in two places:
  - parent selection: which members of current generation will take part in mating
  - survivor selection: which parents and offspring will go into the next generation
- selection operators work on the <u>whole individual</u>
- so they are independent of the chosen problem representation

Fitness Proportionate Selection

- probability for individual i to be selected for mating in a population size μ with FPS is

$$P_{FPS}(i) = f_i \Big/ \sum_{j=1}^{\mu} f_j$$

概率 =　individual fitness = 3, 和总体的fitness的占比 (3+4+5....)

- problems include:
  - premature convergence: one highly fit member can rapidly take over if rest of population is much less fit
    大的fitter在前面, 即使后面有 好的 也选不上了
  - there's a loss of selection pressure when fitnesses are similar, such as at the end of runs　如果分都差不多, 很难选
  - behaves differently if the fitness function is transposed
    分变了, 不好选了
- scaling can be used to fix the last two problems, such as windowing

就是一个放大的思想

Fitness Proportionate Selection: Example

- suppose that `f(A)=2`, `f(B)=3` and `f(C)=5`:

Individual	Fitness for f	Selection Probability for f	Fitness for f+10	Selection Probability for f+10	Fitness for f+100	Selection Probability for f+100	Fitness with Windowing	Selection Probability with Wind.
A	2	0.2	12	0.3	102	0.329	2-1=1	0.14
B	3	0.3	13	0.325	103	0.332	3-1=2	0.29
C	5	0.5	15	0.375	105	0.339	5-1=4	0.57
sum	10	1.0	40	1.0	310	1.0	7	1.0

- we can see that a transposed fitness function changes the selection pressure
- to a point where there's almost no difference between selection probabilities
- windowing can be use to scale the fitness and increase selection pressure:
  - $f'(i) = f(i) - \beta_n$
  - where β_n is worst fitness seen in the last n generations
    - ($\beta_n = 1$ in the example shown above)

Rank-Based Selection

- attempt to remove problems of FPS by basing selection probabilities on relative rather than absolute fitness
- rank population of size μ according to fitness and then base selection probabilities on rank:
  - fittest has rank $\mu - 1$ and worst has rank 0
- this imposes a sorting overhead on the algorithm
- but this is usually negligible compared to the fitness evaluation time

Rank-Based Selection: Linear Ranking

$$P_{lin-rank}(i) = \frac{(2-s)}{\mu} + \frac{2i(s-1)}{\mu(\mu-1)}$$

mu: size of the population
S: 随便选的

- i is the rank
- parameterised by factor s: 1 < s ≤ 2
  - s measures advantage of best individual
- simple 3 member example:

Individual	Fitness	Selection Probability using FPS	Rank (by fitness)	Selection Probability using LR when s=2	Selection Probability using LR when s=1.5
A	2	0.2	0	0	0.167
B	3	0.3	1	0.33	0.33
C	5	0.5	2	0.67	0.5
sum	10	1.0	-	1.0	1.0

Linear Ranking example: s = 1.5, μ = 10

$$P_{lin-rank}(i) = \frac{(2-s)}{\mu} + \frac{2i(s-1)}{\mu(\mu-1)}$$

- suppose population size μ `= 10`
- let's take a middle value for `s` of `1.5`
- note that the first term is always `(1/2)÷10 = 1/20`
- (and we can rewrite this as `9/180` for reasons that will become apparent below)
- then for `i = 0` (least fit member):

```
p = 9/180 + 0 = 9/180
```

- for `i = 1`:

```
p = 9/180 + 2×(1/2)/90 = 1/20 + 1/90
  = (9+2)/180 = 11/180
```

- ...and so on

Individual	Rank (by fitness)	Selection Probability
A	0	9/180
B	1	11/180
C	2	13/180
D	3	15/180
E	4	17/180
F	5	19/180
G	6	21/180
H	7	23/180
I	8	25/180
J	9	27/180
	total:	180/180

So the probability of selection is linearly proportional to rank (which we knew it would be!), with the fittest member 3 times more likely to be selected than the least-fit member.

Linear Ranking example: s = 2, μ = 10

$$P_{lin-rank}(i) = \frac{(2-s)}{\mu} + \frac{2i(s-1)}{\mu(\mu-1)}$$

- now let's go to the extremes of the values for s
- first, suppose that `s = 2`
- note that the first term is always `(2−2)÷10 = 0`
- then for `i = 0` (least fit member):

  `p = 0 + 0/90 = 0/90`
- for `i = 1`:

  `p = 0 + 2/90 = 2/90`
- ...and so on

Individual	Rank (by fitness)	Selection Probability
A	0	0/90
B	1	2/90
C	2	4/90
D	3	6/90
E	4	8/90
F	5	10/90
G	6	12/90
H	7	14/90
I	8	16/90
J	9	18/90
	total:	90/90

So the probability of selection is directly proportional to rank, and we see that the worst member can never be selected.

Linear Ranking example: s = 1, μ = 10

$$P_{lin-rank}(i) = \frac{(2-s)}{\mu} + \frac{2i(s-1)}{\mu(\mu-1)}$$

- note that `s` is always strictly > `1`
- let's see what would happen if we allowed `s = 1`
- then, for all members:

```
p = (2-1)÷10 + (2i×0)/(90)
  = 1/10 + 0 = 1/10
```

Individual	Rank (by fitness)	Selection Probability
A	0	1/10
B	1	1/10
C	2	1/10
D	3	1/10
E	4	1/10
F	5	1/10
G	6	1/10
H	7	1/10
I	8	1/10
J	9	1/10
	total:	10/10

So every member is equally likely to be chosen.

Exponential Ranking

$$P_{\exp-rank}(i) = \frac{1 - e^{-i}}{c}$$

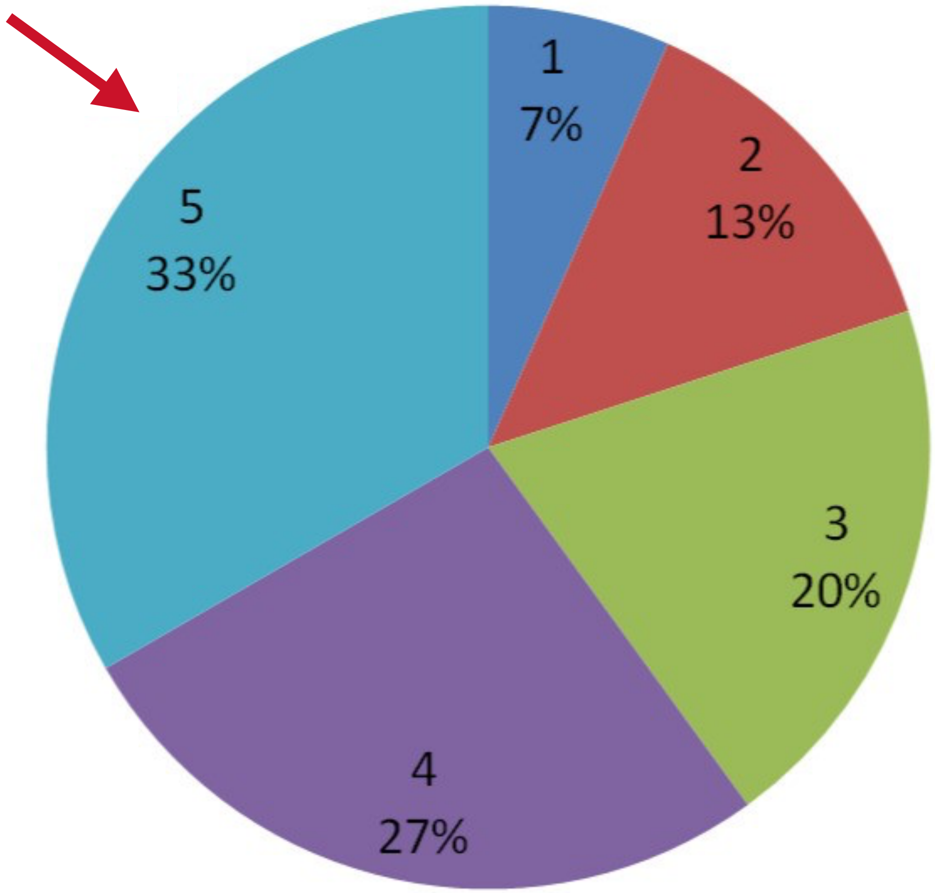
- parameter c is chosen to ensure that the selection probabilities sum to 1.0
- creates more emphasis on creating individuals with above average fitness:

Individual	Fitness	Selection Probability using FPS	Rank (by fitness)	Selection Probability using ER (c = 1.497)
A	2	0.2	0	0
B	3	0.3	1	0.422
C	5	0.5	2	0.578
sum	10	1.0	-	1.0

Roulette Wheel Algorithm

- the roulette wheel algorithm makes λ spins of the wheel
  - one spin for each new individual
- each spin there is one arm pointing to which individual is chosen
- example:

Individual	Fitness	Selection Probability	*Cumulative Probability*
1	1.0	0.07	*0.07*
2	2.0	0.13	*0.20*
3	3.0	0.20	*0.40*
4	4.0	0.27	*0.67*
5	5.0	0.33	*1.00*



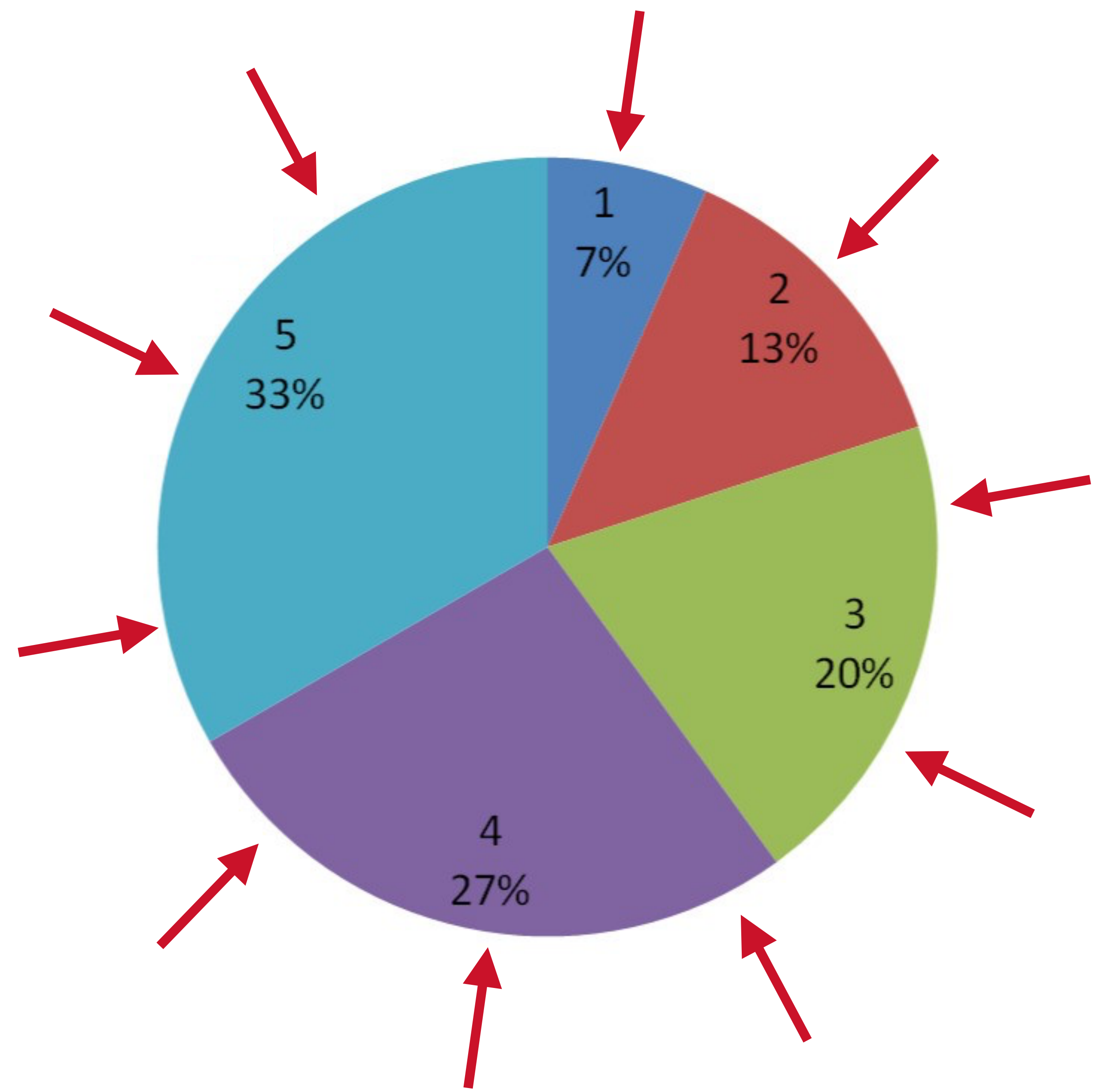
Roulette Wheel Algorithm

```
BEGIN
  // Given the cumulative probability distribution a and
  // assuming we wish to select λ members of the mating pool

set current_member = 1;

WHILE ( current_member ≤ λ ) DO
   Pick a random value r uniformly from [0,1];
   set i=1;

   WHILE( a_i<r ) DO // a_i is the cumulative probability value
      set i=i+1;
   OD

   set mating pool[current_member] = parents[i];
   set current_member = current_member + 1;
OD

END
```

Roulette Wheel Algorithm

stochastic universal sampling (SUS):

- sometimes the roulette wheel algorithm does not give a good sample of the distribution
- so instead of choosing individuals by performing multiple spins…
- SUS uses λ equally-spaced arms and <u>spins the wheel just once</u>
- this brings the actual number of times each parent is chosen much closer to the expected value



λ = number of children to be created (10 here)

Parent Selection: Tournament Selection

- all the previous methods above rely on global population statistics
- this could cause a bottleneck in computation
  - especially on parallel machines, or with a very large population
- they rely on the presence of an external fitness function which might not exist:
  - for example, evolving game players
- one idea for a procedure that only uses local fitness information:
  - pick `k` members at random then select the best of these
  - repeat to select more individuals

Parent Selection: Tournament Selection

- the probability of selecting individual `i` will depend on:
  - rank of `i`
  - size of sample `k`
    - higher `k` increases selection pressure
  - whether or not contestants are picked with replacement
    - picking without replacement increases selection pressure
  - whether or not the fittest contestant always wins (deterministic), or wins with probability `p` (stochastic)

Parent Selection: Tournament Selection

```
BEGIN
  // Assume we wish to select λ members of a pool of μ
  // individuals

set current_member = 1;

WHILE ( current_member ≤ λ ) DO
   Pick k individuals randomly, with or without replacement;
   Compare these k individuals and select the best of them;
   Denote this individual as i;

   set mating pool[current_member] = i;
   set current_member = current_member + 1;
OD

END
```

Case Study: Iterated Prisoner's Dilemma

	B stays silent (cooperates)	B betrays A (defects)
A stays silent (cooperates)	Both serve 1 year	A serves 3 years, B goes free
A betrays B (defects)	A goes free, B serves 3 years	Both serve 2 years

- iterated version: two players play a series of games against each other
- allows opportunity for evolution of strategies:
    - when to cooperate, when to defect, how to react
- easy to determine the fitness of individuals by playing them against each other
- and that's exactly what's been done - many times!
- these links introduce the problem:
    - <u>The Iterated Prisoner's Dilemma</u>
    - <u>Axelrod: The Evolution of Cooperation</u>
- …and there's lots more in the Resources section of Brightspace

Parent Selection: Uniform

- effectively no selection pressure at this stage
- leaves that job to the survivor selection strategy
- over-selection: a variant where the population is split into two groups based on fitness
    - top `x%` in first group
    - bottom `100-x%` in second group
- more selections chosen from the first group than the second
- typically an 80/20 split

Survivor Selection

- this is managing the process of reducing the working memory of the EA from:
    - a set of μ parents and λ offspring
        - to
    - a set of μ individuals forming the next generation
- the parent selection mechanisms can also be used for selecting survivors
- survivor selection can be divided into two approaches:
    - age-based selection
        - fitness is not taken into account
        - in steady state GA can implement as 'delete-random' (not recommended) or as first-in-first-out (also known as 'delete-oldest')
    - fitness-based replacement

Fitness Based Replacement

elitism:

- always keep at least one copy of the fittest solution (or top % of solutions) so far
- widely used in both generational and steady state population models

GENITOR (‘delete-worst’):

- from Whitley’s original steady-state algorithm
- rapid takeover - can lead to rapid convergence
  - so best used with a large population, or with a ‘no duplicates’ policy

Fitness Based Replacement

round-robin tournament:

- μ parents, λ offspring
- pairwise competitions in round-robin format:
  - every pair of individuals compared to each other
  - count number of 'wins' for each individual
  - μ individuals with most wins selected to form the new population

Fitness Based Replacement

(μ, λ)-selection

- based on the set of children only ($\lambda > \mu$)
- choose best μ

$(\mu + \lambda)$-selection

- based on the set of parents and children
- choose best μ

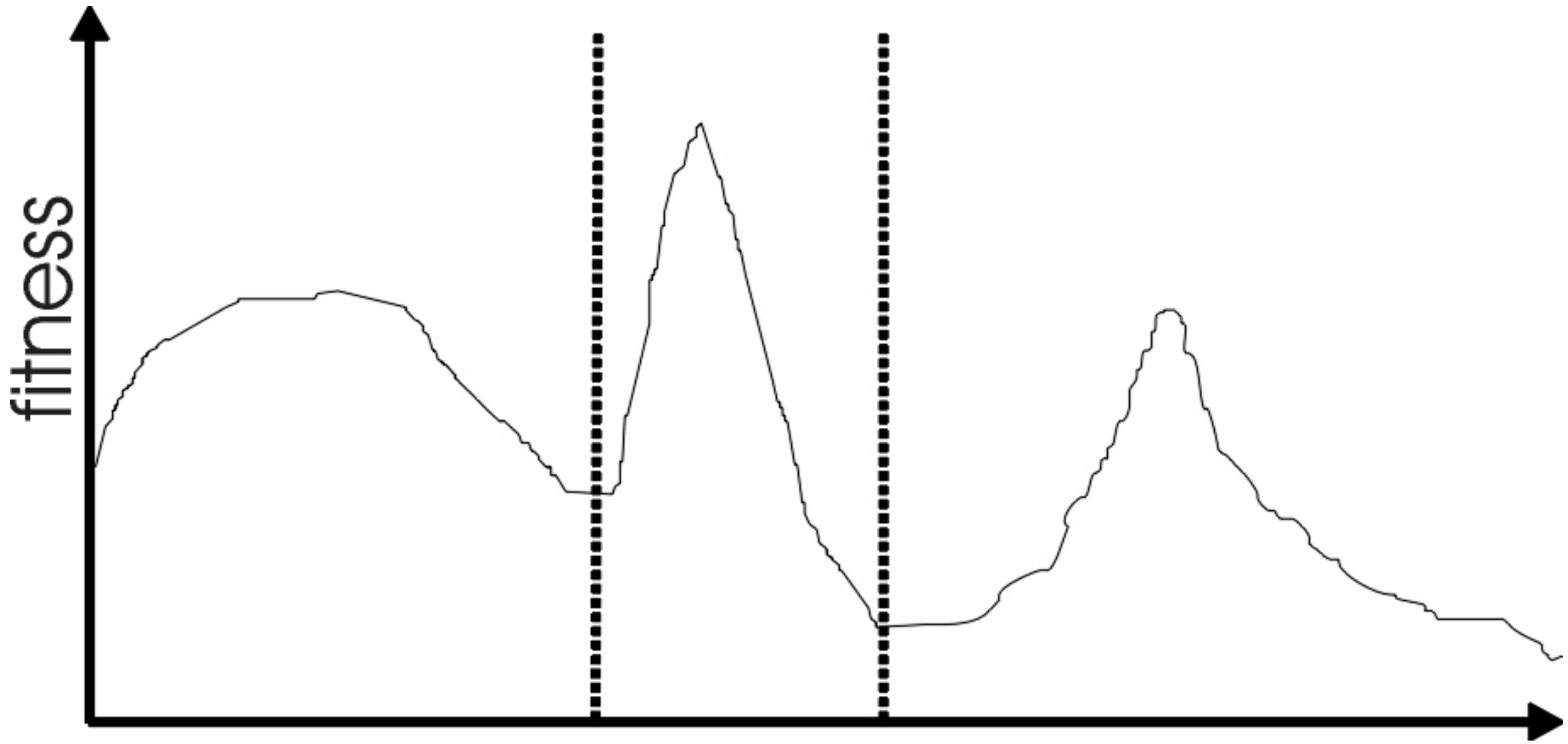
- often (μ, λ)-selection is preferred because:
  - better ability to escape local optima
  - better ability to follow moving optima
  - when using the $(\mu + \lambda)$ strategy, bad σ values can survive in $\langle x, \sigma \rangle$ for too long if their host x is very fit
- $\lambda = 3.\mu$ is usually good setting

Selection Pressure

- takeover time τ^* is a measure to quantify the selection pressure
- the number of generations it takes until the application of selection completely fills the population with copies of the best individual
- for fitness proportional selection in a genetic algorithm the takeover time is $\lambda \ln(\lambda)$

Multimodality

- most interesting problems have more than one locally optimal solution:



Multimodality: Genetic Drift

- finite population with global mixing and selection eventually convergence around one optimum
- why?
- often might want to identify several possible peaks
  - perhaps to make it easier to adapt the solution to other problems
- sub-optimum can be more attractive
  - perhaps aesthetically

Approaches for Preserving Diversity: Introduction

explicit vs implicit:

- implicit approaches:
  - impose an equivalent of geographical separation
  - impose an equivalent of speciation
- explicit approaches:
  - make similar individuals compete for resources (fitness)
  - make similar individuals compete with each other for survival

Approaches for Preserving Diversity: Introduction

different spaces:

- genotype space
    - set of representable solutions
    - ‘decision space’
- phenotype space
    - the end result
    - neighbourhood structure may bear little relation with genotype space
    - ‘solution space’
- algorithmic space
    - •equivalent of the geographical space on which life on earth has evolved
    - structuring the population of candidate solutions
    - such as splitting a population over a number of different processors or cores
- any technique that maintains diversity in the population based on the measure of some 'distance' between the population members is called a niching technique

Explicit Approaches for Preserving Diversity: Fitness Sharing

(Goldberg & Richardson)

- a type of niching where the fitness of each individual is scaled based on its proximity to others
- so good solutions in densely populated regions are given a lower fitness value than comparably good solutions in sparsely populated regions
- which lowers their chances of selection
- and therefore increases the chances of maintaining a population of solutions occupying several niches

Explicit Approaches for Preserving Diversity: Fitness Sharing

- the distance between individuals can be calculated in several ways, based either on values in:
  - genotype space
    - usually Hamming distance
  - phenotype space
    - usually Euclidian distance
  - or both

Fitness Sharing: Pseudocode

```
define:
minimum_distance:
   - a measure of niche size
   -any solution j closer to current solution i than minimum_distance will
    share fitness
shareParameter:
   - a parameter that determines how much influence sharing has on the
fitness
     value


for each individual i in the population:
   denominator = 1
   for each individual j in the population:
      calculate the distance between i and j
      if distance < minimum_distance: // if share the same niche
         denominator = denominator + (1 - distance/share_parameter)
   fitness[i] = fitness[i]/denominator
```

Explicit Approaches for Preserving Diversity: Crowding

(de Jong)

- attempts to distribute individuals evenly amongst niches
- ensures that new individuals replace similar members of the population
- so offspring replace the parents that they are most similar to

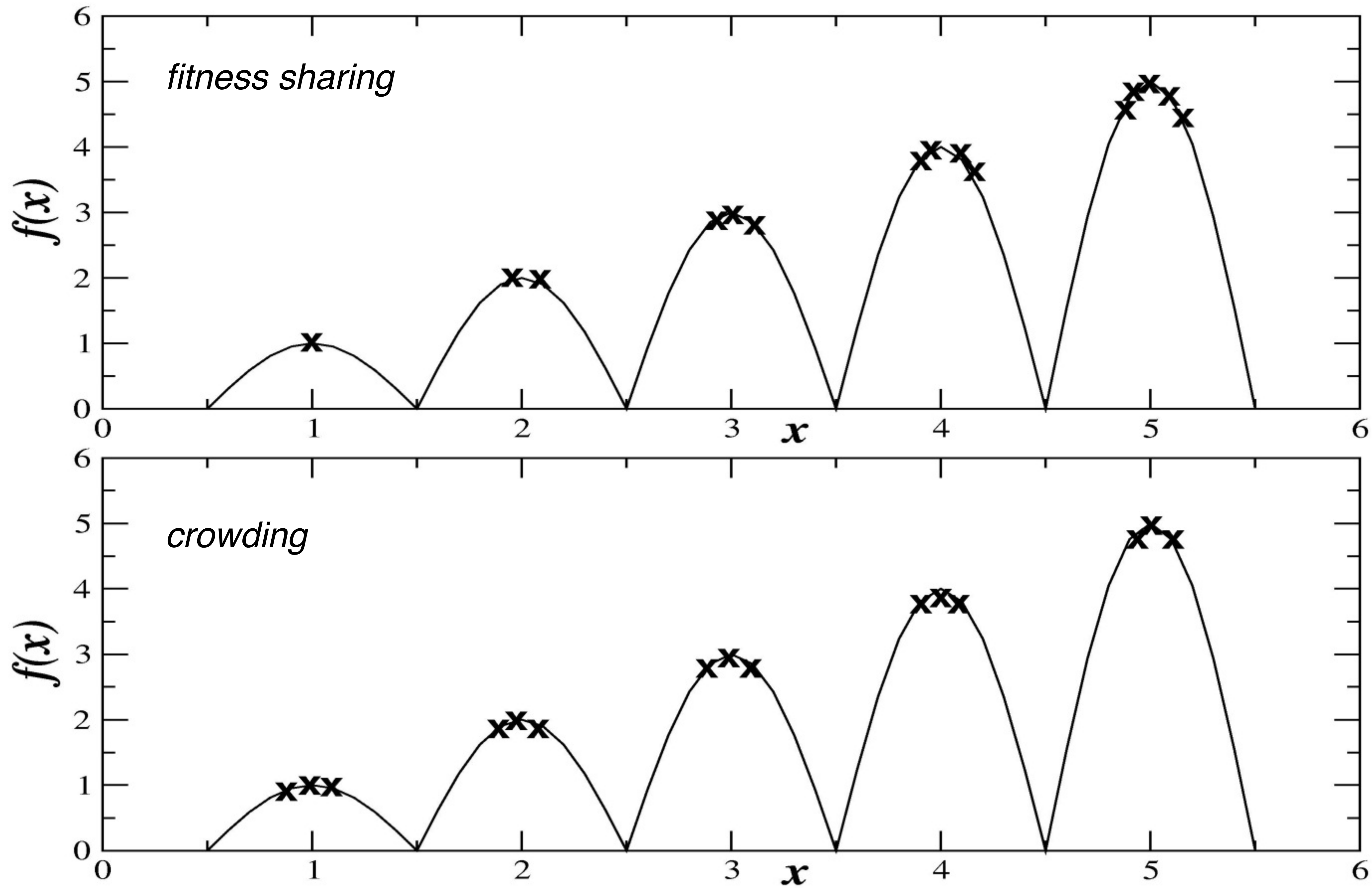
Explicit Approaches for Preserving Diversity: Crowding

deterministic crowding (Mahfoud):

1. parent population is randomly paired
2. each pair produces two offspring via recombination
3. offspring are mutated and evaluated
4. pairwise distances between parents and offspring are calculated
5. each offspring competes for survivial against one parent for survival, using the parent-offspring pairings that minimize the overall distances between each pair:

$$d(p_1,o_1) + d(p_2,o_2) < d(p_1,o_2) + d(p_2,o_1)$$

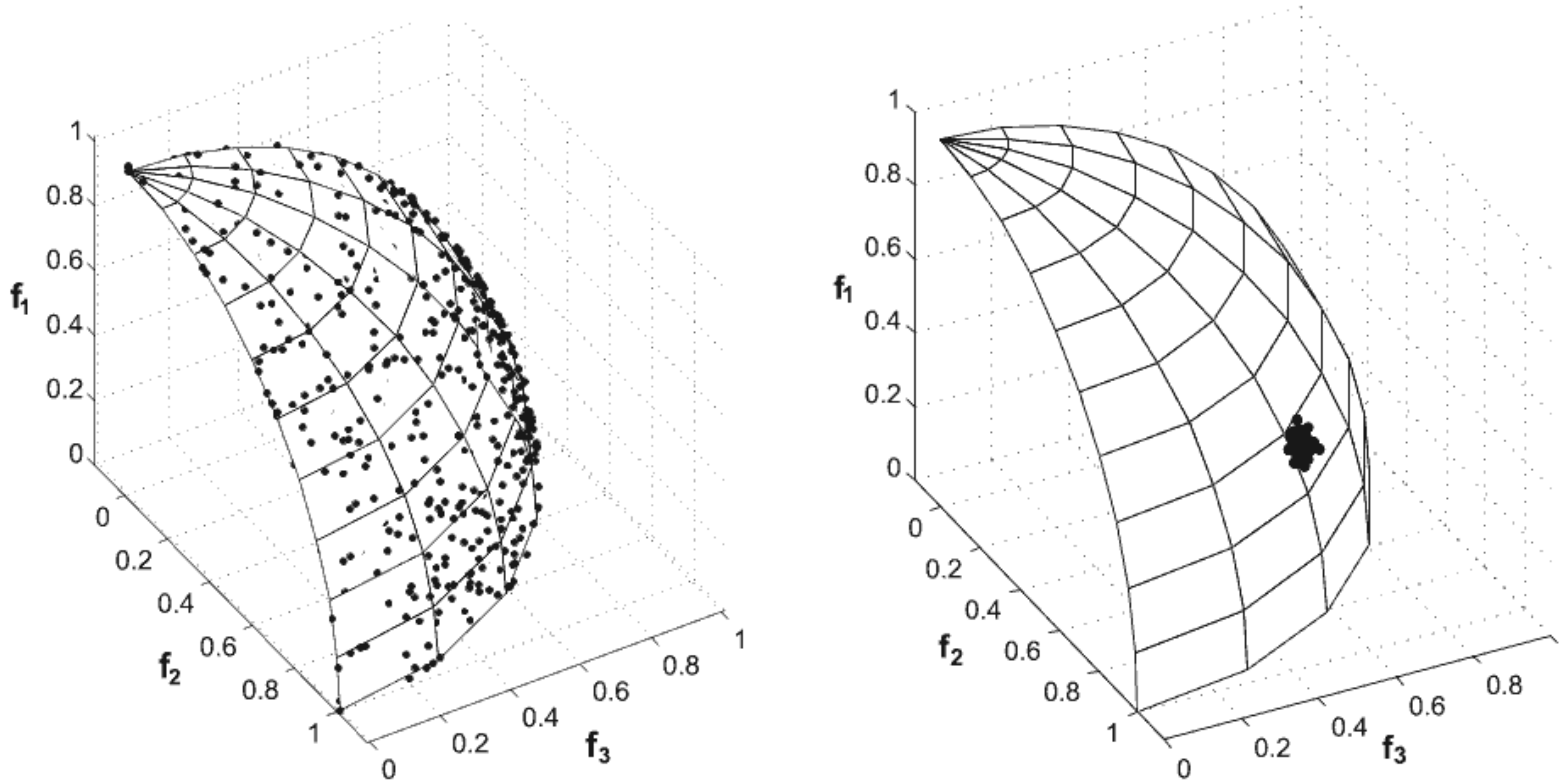
Explicit Approaches for Preserving Diversity: Crowding or Fitness sharing?



observe the number of individuals per niche…

An Aside: Maintaining Population Diversity for Multi-objective Problems

- ‘pareto fronts’ with (a) and without (b) fitness sharing



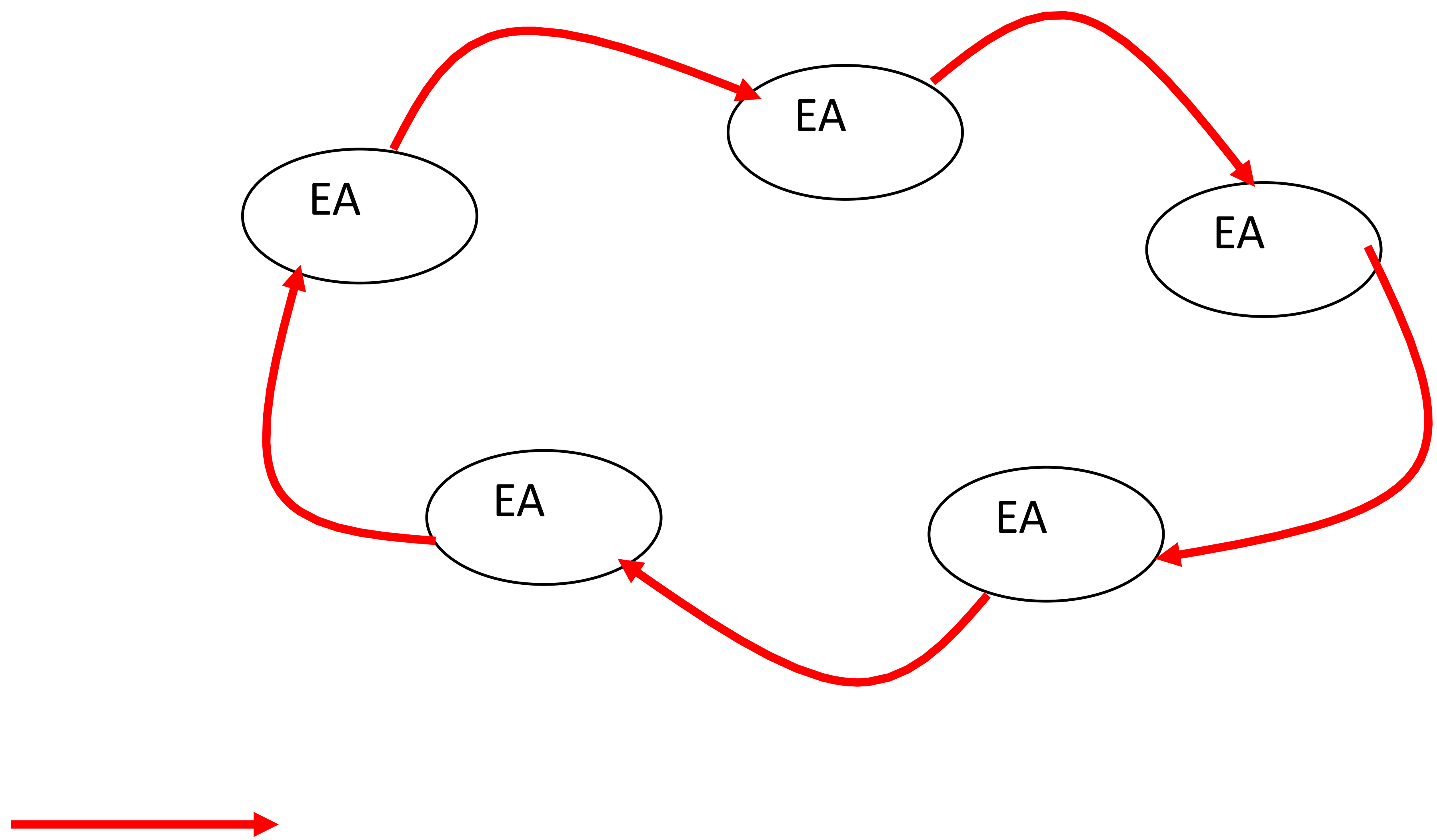
(a) (b)

Implicit Approaches for Preserving Diversity: Automatic Speciation

use mating restrictions:

- either only mate with genotypically / phenotypically similar members
    - or
- add tags to problem representation
    - extra genes (one per genotype) that acts as a label for which species the genotype belongs to
    - initially randomly set
    - subject to recombination and mutation
    - when selecting partner for recombination, only pick members with a good match

Implicit Approaches for Preserving Diversity: Island Model: Parallel EAs



periodic migration of individual solutions between populations

Implicit Approaches for Preserving Diversity: Island Model: Parallel EAs

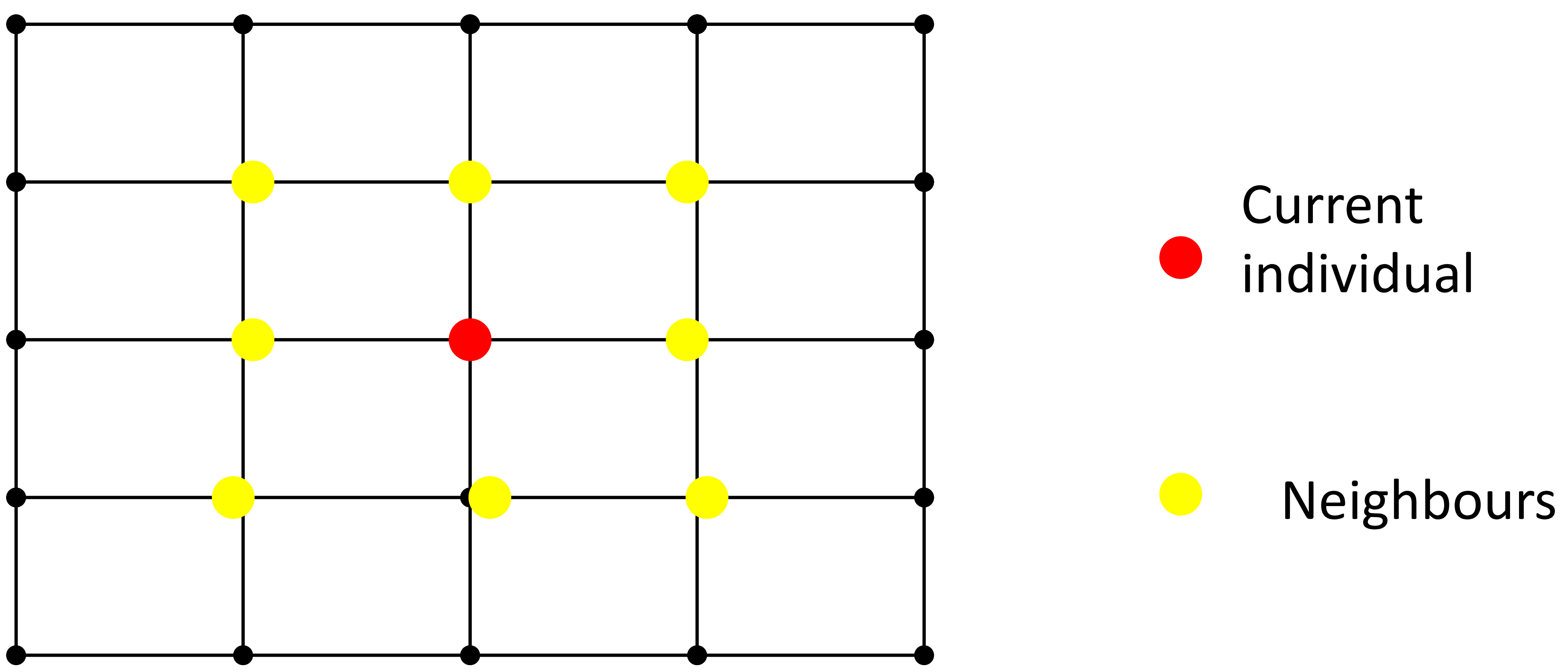
- run multiple populations in parallel
- after a (usually fixed) number of generations - an epoch - exchange individuals with neighbours
- repeat until ending criteria met
- partially inspired by parallel/clustered systems

Island Model: Parameters

- how often should individuals be exchanged between populations?
  - too quick and all sub-populations converge to same solution
  - too slow and waste time
  - most authors use range of 25 to 150 generations
  - can do it adaptively:
    - stop each pop when no improvement for (say) 25 generations
- how many and which individuals should be exchanged?
  - usually around 2 to 5, but depends on population size
  - copied vs moved
  - Martin et al found that better to exchange randomly selected individuals than best
- can have a flexible approach: operators can differ between the sub-populations

Implicit Approaches for Preserving Diversity: Cellular EAs

- impose spatial structure - usually a grid



Implicit Approaches for Preserving Diversity: Cellular EAs

- selection and replacement take place using the concept of neighbourhoods
- the use of neighbourhoods for parent and survivor selection leads to different parts of grid searching different parts of space
- while, because neighbourhoods overlap, good solutions diffuse across grid over a number of generations

Implicit Approaches for Preserving Diversity: Cellular EAs

example of use:

- equivalent of 1 generation is:
    - pick individual in population at random
    - pick one of its neighbours using roulette wheel
    - crossover to produce 1 child, and mutate
    - replace individual with child if fitter
    - circle through population until done

Reading & References

- slides based on and adapted from, Chapter 5 (and slides) of Eiben & Smith's *Introduction to Evolutionary Computing*
- see the Resources section of Brightspace for wider reading
- E. Alba and B. Dorronsoro. *Cellular Genetic Algorithms. Computational Intelligence and Complexity.* Springer, 2008.
- G. Luque and E. Alba. *Parallel Genetic Algorithms, volume 367 of Studies in Computational Intelligence.* Springer, 2011.