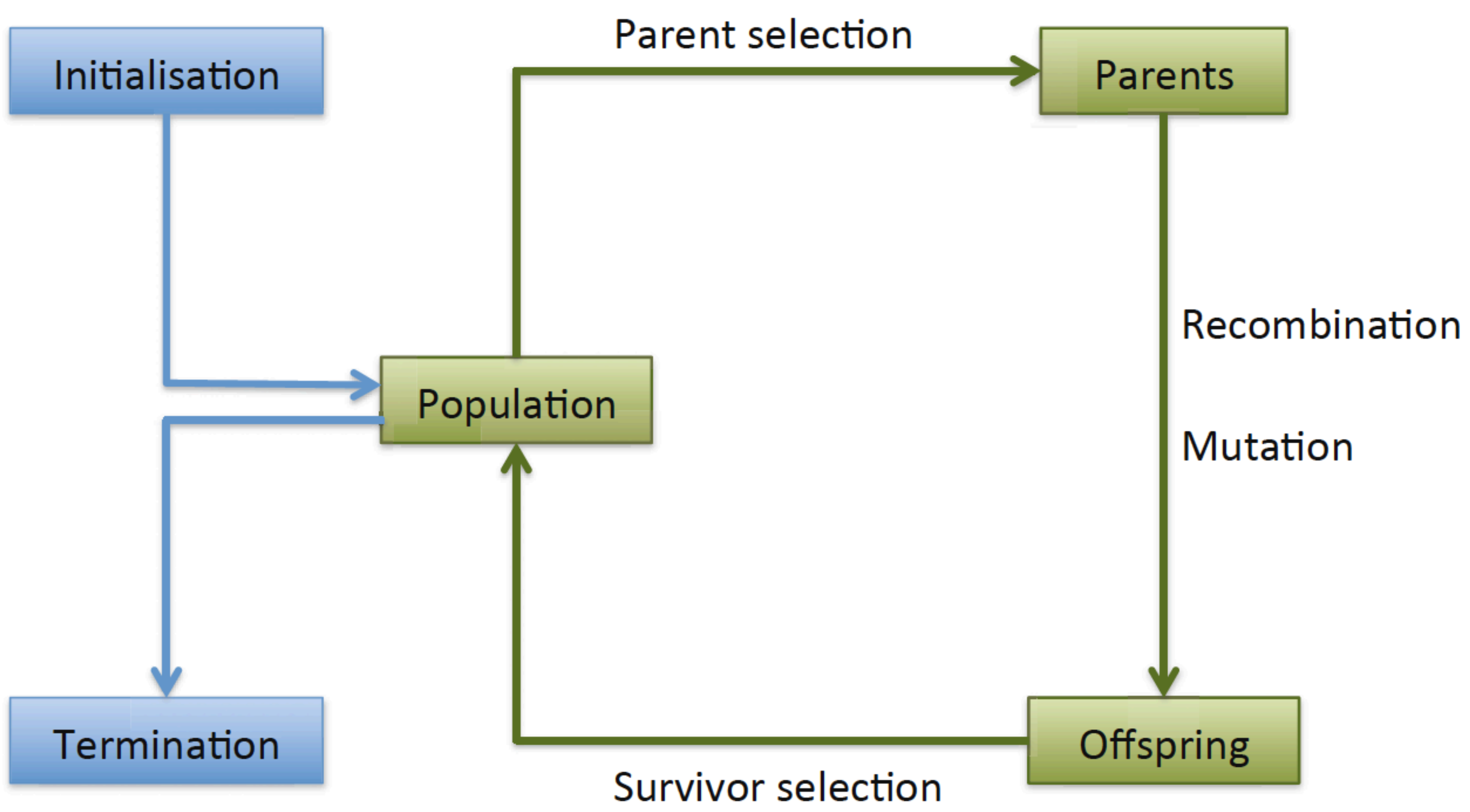
8: Selection and Population Management

- Population management models
  - generational, steady-state
- Parent selection
  - fitness proportional selection, ranking selection, tournament selection, uniform selection
- Survivor selection
  - age-based, fitness-based
- Textbook Chapter 5.1 - 5.3

Prof. Ting Hu, School of Computing, Queen’s University

Recall: general scheme of EAs



Population management

- SGA uses a *generational* model:
  - each individual survives for exactly one generation (popsize μ)
  - a mating pool of μ parents create λ offspring
  - the entire set of parents is replaced by the offspring (usually $\mu = \lambda$)
- *Steady-state* model:
  - $\lambda(< \mu)$ offspring are generated per iteration
  - λ members of population replaced in each iteration

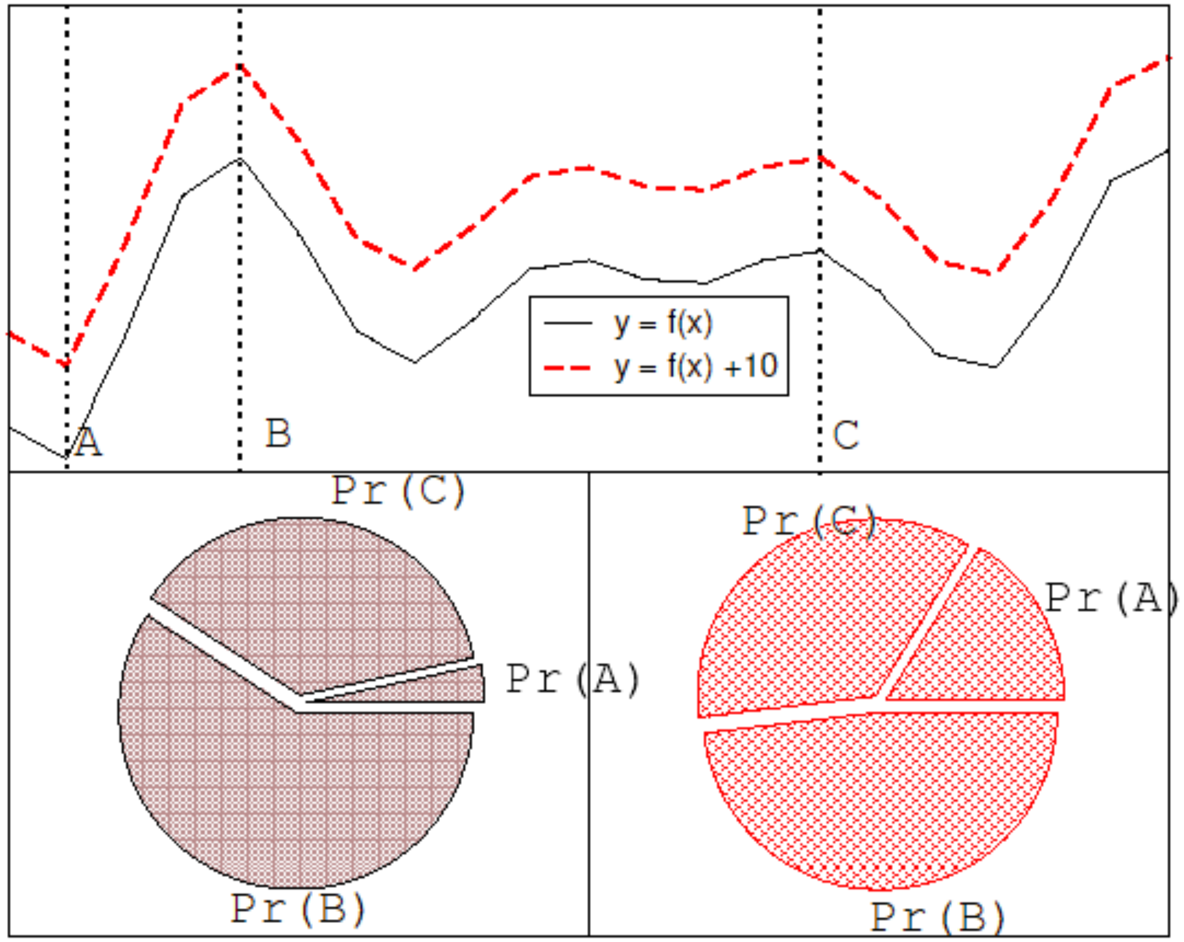
Fitness-based competition

- Selection can occur in two places:
  - select from current generation to take part in reproduction (**parent selection**)
  - select from parents + offspring to go into next generation (**survivor selection**)
- Selection operators work on whole individuals
  - i.e., they are representation-independent
- Steps
  - operators: define selection probabilities
  - algorithms: define how probabilities are implemented

Parent selection: fitness-proportional selection (FPS)

- Probability for individual *i* to be selected for mating in a population of size μ with FPS is $P_{FPS}(i) = f_i \Big/ \sum_{j=1}^{\mu} f_j$
- Issues with FPS:
  - One highly fit member can rapidly take over if rest of population is much less fit — premature convergence
  - At the end of runs when fitness levels are similar, lose **selection pressure**
  - Highly susceptible to function transposition

Function transposition on FPS



Individual	Fitness for f	Sel. prob. for f	Fitness for $f+10$	Sel. prob. for $f+10$	Fitness for $f+100$	Sel. prob. for $f+100$
A	1	0.1	11	0.275	101	0.326
B	4	0.4	14	0.35	104	0.335
C	5	0.5	15	0.375	105	0.339
Sum	10	1.0	40	1.0	310	1.0

Parent selection: fitness-proportional selection (FPS)

- Scaling can fix the second and third issues
  - windowing $f'(i) = f(i) - \beta^t$

    where β is worst fitness in this (or last *n*) generation(s)
  - sigma scaling $f'(i) = max(f(i) - (\bar{f} - c\sigma_f), 0)$

    where *c* is a constant, usually 2.0

Parent selection: rank-based selection

- Attempt to fix issues of FPS by basing selection probabilities on *relative* rather than *absolute* fitness
- Rank population according to fitness and then base selection probabilities on rank where fittest has rank $\mu - 1$ and worst rank 0
- Imposes a sorting overhead on the algorithm, but usually negligible compared to the fitness evaluation time

Linear ranking

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

- Parameterized by factor *s*: 1.0 < *s* <= 2.0
  - measures advantage of best individual

- Simple 3-member example

Individual	Fitness	Rank	P_{selFP}	$P_{selLR} \quad (s=2)$	$P_{selLR} \quad (s=1.5)$
A	1	0	0.1	0	0.167
B	4	1	0.4	0.33	0.33
C	5	2	0.5	0.67	0.5
Sum	10		1.0	1.0	1.0

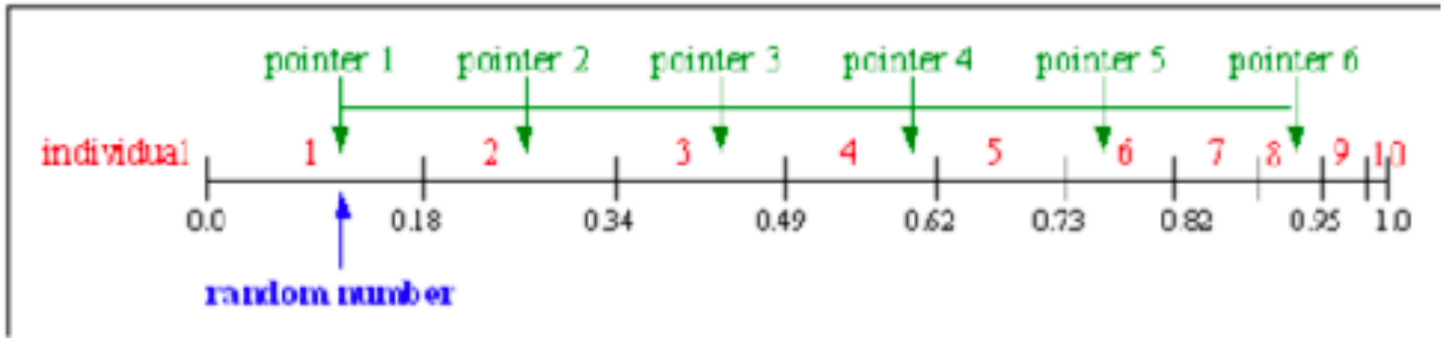
Exponential ranking

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

- Linear ranking is limited to selection pressure
- Exponential ranking can allocate more than two copies to fittest individual
- Normalize constant factor *c* is chosen according to population size -> the sum of the probabilities is unity

Implementation of FPS

- Roulette wheel algorithm:
  - Given a probability distribution, spin a 1-armed wheel λ times to make λ selections
  - No guarantees on actual numbers of copies
  - Strong individuals have a better chance, but everyone has some chance
- Multi-pointer selection MPS (stochastic universal sampling):
  - λ evenly spaced arms on wheel and spin once
  - equally spaced the selected samples with distance $1/\lambda$



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
      set i = i + 1;
    OD
    set mating_pool[current_member] = parents[i];
    set current_member = current_member + 1;
  OD
END
```

Multi-pointer selection (MPS)

```
BEGIN
  /*  Given the cumulative probability distribution a */
  /*  and assuming we wish to select λ members of the mating pool */
  set current_member = i = 1;
  Pick a random value r uniformly from [0, 1/λ];
  WHILE ( current_member ≤ λ ) DO
    WHILE (  r ≤ a[i] ) DO
      set mating_pool[current_member] = parents[i];
      set r = r + 1/λ;
      set current_member = current_member + 1;
    OD
    set i = i + 1;
  OD
END
```

Tournament selection

- FPS methods above rely on global population statistics
  - could be a bottleneck especially on parallel machines, very large population
  - relies on presence of external fitness function which might not exist, e.g. evolving game players or evolutionary design and art
- Informal procedure
  - pick *k* members at random then select the best
  - Repeat to select more individuals

Tournament selection

- Probability of selecting *i* will depend on:
  - rank of *i*
  - tournament size *k*
    - higher *k* increases selection pressure
  - whether contestants are picked with replacement
    - with replacement the worst has a chance to get selected
  - whether fittest contestant always wins (deterministic) or this happens with probability *p*

Implementation of 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
```

Uniform parent selection

$$P_{uniform}(i) = 1/\mu$$

- Each individual has the same chance to be selected as a parent
- Would need to couple with a strong fitness-based survivor selection

Overselection

- If the potential search space is enormous
- Use a extremely large population
    - to avoid missing promising regions in the initial random generation
    - to maintain the diversity needed to support exploration
- Overselection
    - the population is first ranked by fitness and then divided into two groups
    - the top x% and the remaining (100-x)%
    - 80% of parents chosen from the first group and 20% from the second
    - decrease x when population size increases

Population size	Proportion of population in fitter group (x)
1000	32%
2000	16%
4000	8%
8000	4%

Survivor selection

- Managing the process of reducing the working memory of the EA from a set of μ current individuals and λ offspring to a set of μ individuals forming the next generation
- 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
    - steady-state GA uses first-in-first-out (a.k.a. delete-oldest)
    - needs strong parent selection and less disruptive variation operators
  - Fitness-based
- Elitism
  - often used in conjunction with age-based and stochastic fitness-based survivor selection

Fitness-based survivor selection

- ($\mu + \lambda$) selection
  - pool all μ current generation individuals and λ offspring
  - rank them
  - pick the top μ individuals to enter next generation
- Replacement (usually $\mu > \lambda$)
  - use λ offspring to replace the worst λ current generation individuals
- Round-robin tournament
  - in pairwise tournaments, each individual is evaluated against q others randomly chosen from the merged parents and offspring population
  - μ individual with the greatest number of wins are selected as survivors, $q = 10$
- Random uniform