Artificial Life

Lecture 4 : The black art of GA's and making it easier

Chris Johnson

This Lecture

- Searchscapes
- Microbial GA (simplifying the implementation)
- Diversity maintenance

Reading

- Harvey, I. (2009) The Microbial Genetic Algorithm In G. Kampis et al (Eds.) Proceedings of the Tenth European Conference on Artificial Life, Springer LNCS. http://www.cogs.susx.ac.uk/users/inmanh/MicrobialGA_ECAL2009.pdf

Population Based Evolutionary Algorithm

NoGenes, *NoIndividuals*, *NoGenerations*

Initialise population (matrix, *Pop*)

Calculate Fitness (vector, *Fit*)

for i=1:*NoGenerations* %(or for some termination condition)

While(*New_Pop* != Full)

- Select parents proportional to fitness.
- Crossover parents to create children.
- Mutate children.
- Calculate fitness of children New_Fit
- Add children to New_Pop

end

Pop = *New_Pop*;

Fit = New_Fit

end

Why do GA’s work?

- Why is finding a word in a dictionary easy?
- Because it is organised (A..Z), allowing for us to make guesses, and to use those guesses to correct where we look next..

hair	hat	hip
hall	hay	horn
hammer	head	horse
hamster	hear	hot dog
hand	heart	house
happy	help	hug
hard	hen	

Patterns in Search Space

- If the words were randomly distributed, you'd struggle!
- So, if a search-space is organised, we can spend less effort searching it...

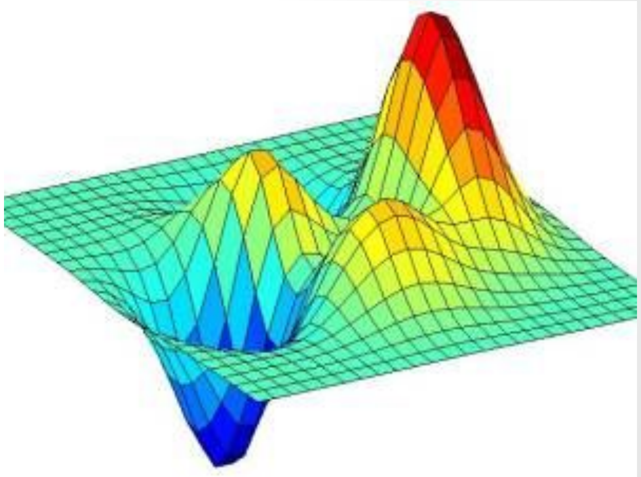
Truck	Grapes	Bulb
Balloon	Blocks	Door
Music	Rectangle	Blocks
Note	Submarine	Rectangle
Octopus	Deer	Guitar
Telephone	Igloo	Engagement Ring
Stop Sign	Dog	Dragon
Apple	Flower	Bee
Tree	Train	Rain
Cat	Wand	Bow and Arrow
Shoe	Light	Star

Patterns in Search Space

- Genetic algorithms **taking advantages of these kinds of trends in search space**, minimising the number of samples we have to take, by assuming that there are trends or patterns in the search space (think of the Dictionary example)
- In particular, to work effectively GAs require similar genotypes to produce similar phenotypes with similar fitness
- many paths leading from low-fitness genotypes to high-fitness genotypes

Fitness Landscape

- We can visualise the fitness landscape for a hypothetical 2D (two-gene) problem
- 2-genes (X, Z)
- Height (Y) is how fit that combination of genes is
- Red = high fitness
- Blue = low fitness
- Cyan = average fitness



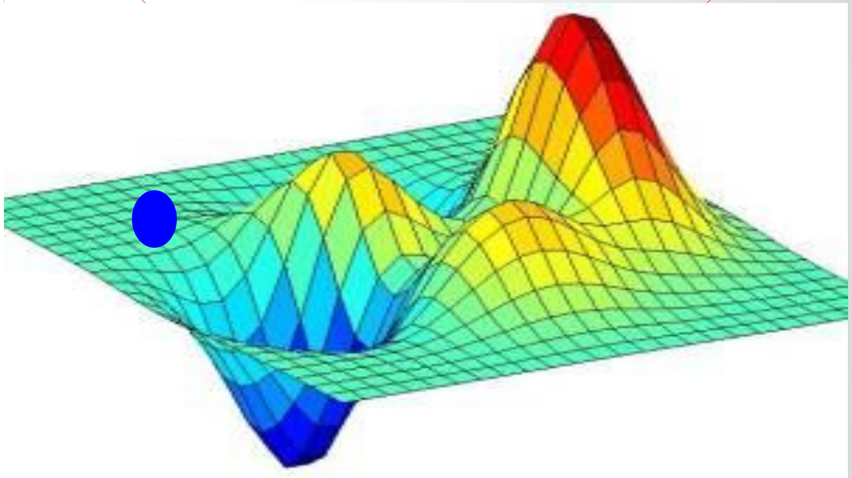
Mutation as Motion

Motion

- "Mutation" is sort of like moving through the fitness landscape (different ways to do this..more later)

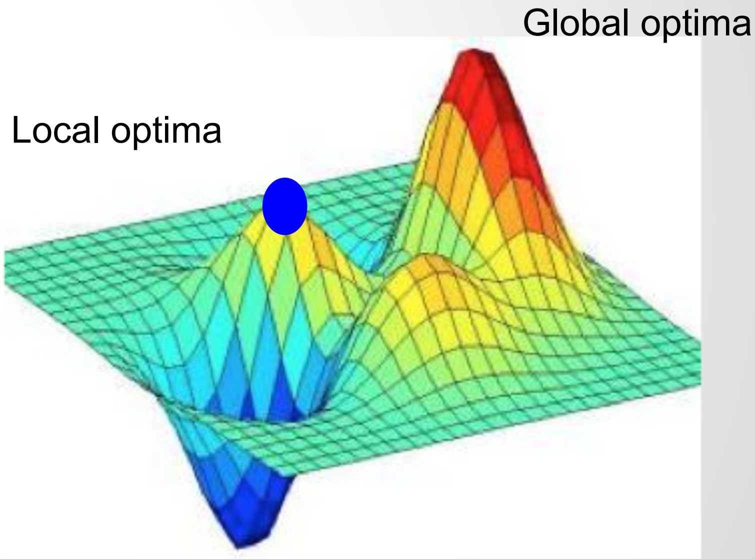
Distance

- Points that are close together in the fitness landscape are those for which it takes few mutations to move between



A Simple Hillclimber

1. Change (mutate) a by a random amount (small changes more likely)
2. Measure fitness. If it is worse, undo the change.
3. Goto 1

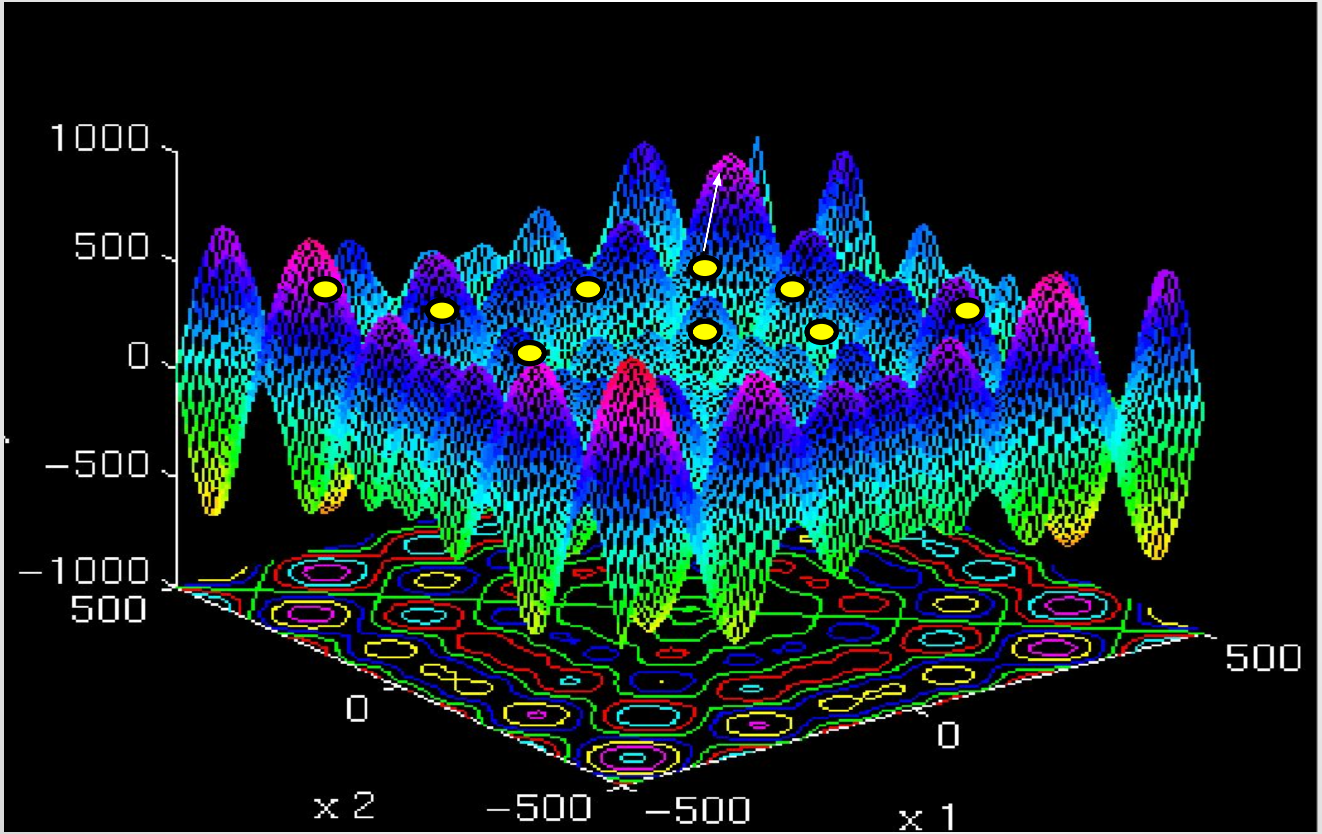


Benefits of a population

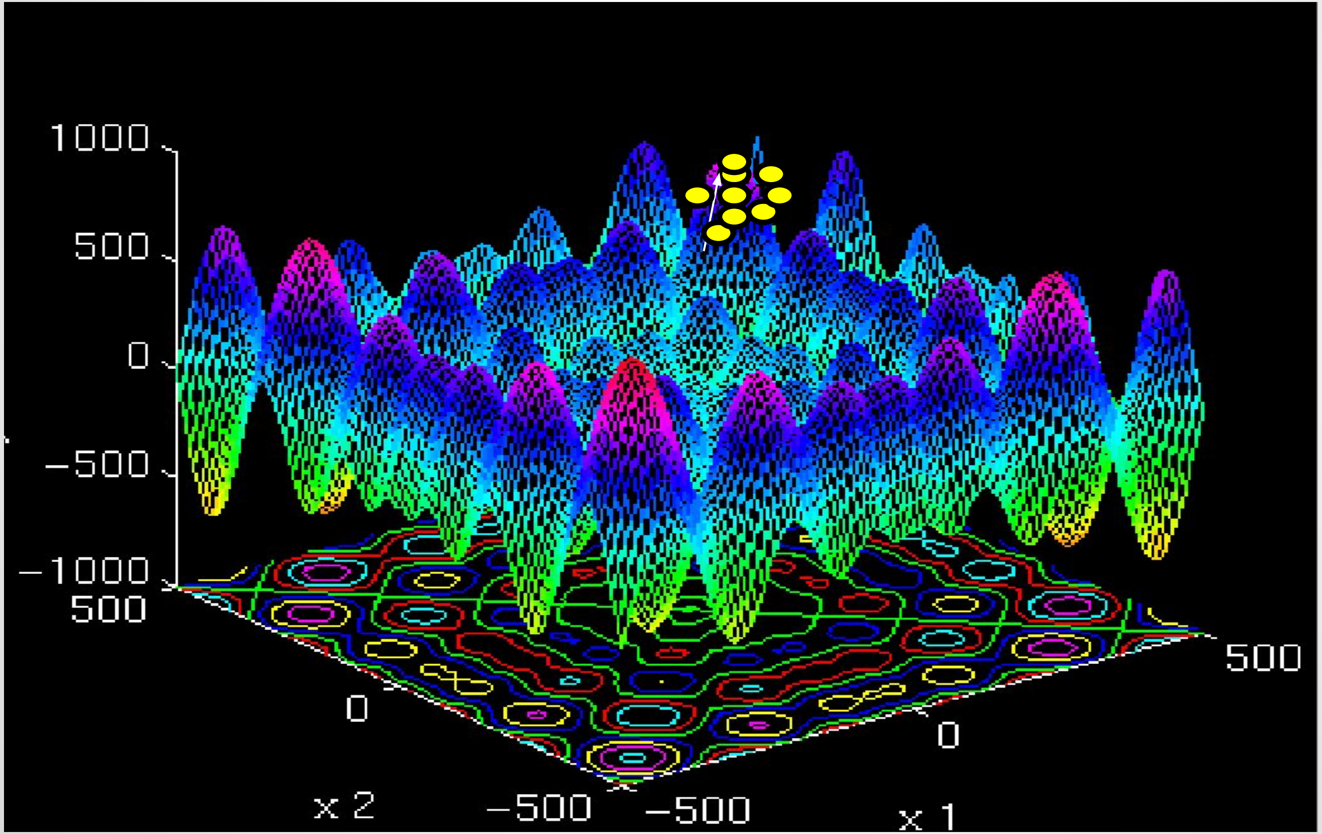
- Evolutionary algorithms
  - Like a hill-climber but with a population
  - Make many random individuals
  - Some individuals might be luckier than others and we can preferentially focus on the better ones

  - Furthermore we can generate new solutions from a bit of one combined with a bit of another (using crossover). See lecture on compositional evolution.

Benefits of a population



Benefits of a population



Diversity Maintenance

- The management of diversity is central to successful GA optimization
- Align with balancing central axioms of evolution
  - HEREDITY - offspring are (roughly) identical to their parents.
  - VARIABILITY - except not exactly the same, some significant variation.
- Mutation increases diversity allowing exploration but destroys heredity inhibiting exploitation
- Crossover tends to decrease diversity too

Diversity Maintenance:Black Arts

- Population size
  - Too big?
    - waste of time
  - Too small?
    - no useful diversity in population
- Mutation rate
  - Too low?
    - too little innovation
  - Too high?
    - too much destruction

Population Based Evolutionary Algorithm

NoGenes, *NoIndividuals*, *NoGenerations*

Initialise population (matrix, *Pop*)

Calculate Fitness (vector, *Fit*)

for i=1:*NoGenerations* %(or for some termination condition)

    While(*New_Pop* != Full)

- Select parents proportional to fitness.
- Crossover parents to create children.
- Mutate children.
- Calculate fitness of children
- Add children to New_Pop

    end

    Pop = *New_Pop*;

end

- **This is much too complicated**
- **Need two arrays**
- **Roulette wheel selection is a pain.**

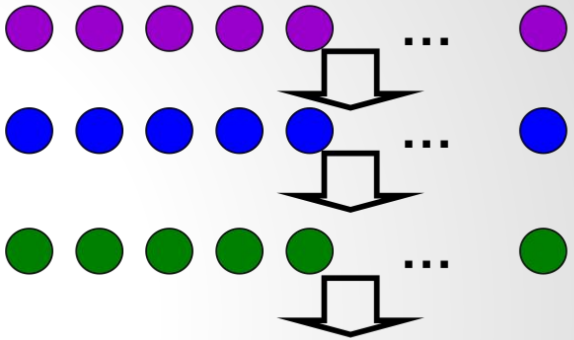
Get rid of roulette wheel selection: Tournament Selection

```
P1=Pop(floor(rand*P+1),:)
P2=Pop(floor(rand*P+1),:)
If (fitness(P1)>fitness(P2))
    Winner =  P1;
    Loser = P2;
else
    Winner =  P2;
    Loser = P1;
end
```

- More generally, pick k individuals and return individual with greatest fitness

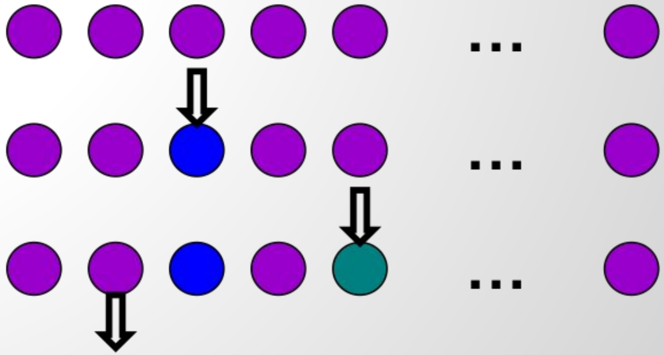
Get rid of old and new population: Steady State Genetic Algorithm

Instead of a Generational GA, replacing all n at the same time

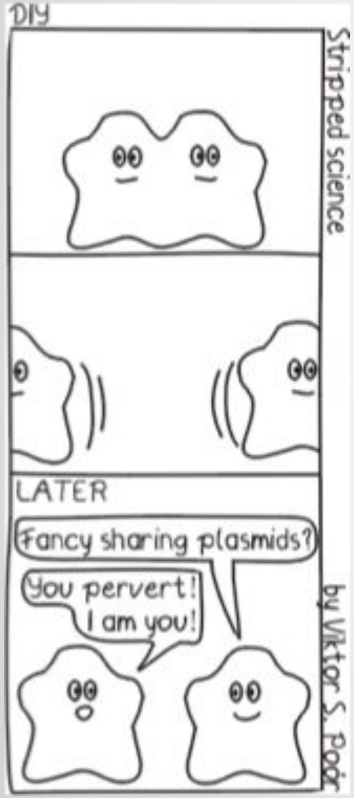


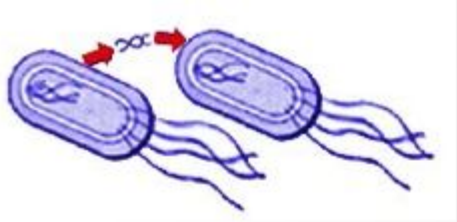
You can just produce one new offspring at a time, replacing one.

(Repeat n times for the equivalent of one generation)



Adding Crossover: Microbial Sex





Instead of “Let’s make babies !”

It is

“Want to share some of my genes?”

Horizontal Gene Transmission

- The movement of genetic material between two organisms. Once incorporated it is then ‘vertically’ inherited.
- Only work in organisms with relatively close genetic makeup.
- Mainly Eukaryotes rather prokaryotes
- Antibiotic resistance genes on plasmids

  Ochman, Lawrence, and Groisman, Nature 405:299-304

Microbial Genetic Algorithm – the algorithm

Pick two genotypes at random

- Compare scores -> Winner and Loser
- Go along genotype, at each locus with some prob copy from Winner to Loser (overwrite)
- with some prob mutate that locus of the Loser. So ONLY the Loser gets changed, (giving a version of Elitism for free!)

```
for i=1:NoGenes
    If(rand < Pc)
        Loser(i) = Winner(i);
    end
    if(rand < Pm)
        Loser(i) = flip(Loser(i))
    End
end
```

Microbial GA (Psuedocode)

NoGenes, *NoIndividuals*, *NoTournaments*

Initialise population (matrix, *Pop*)

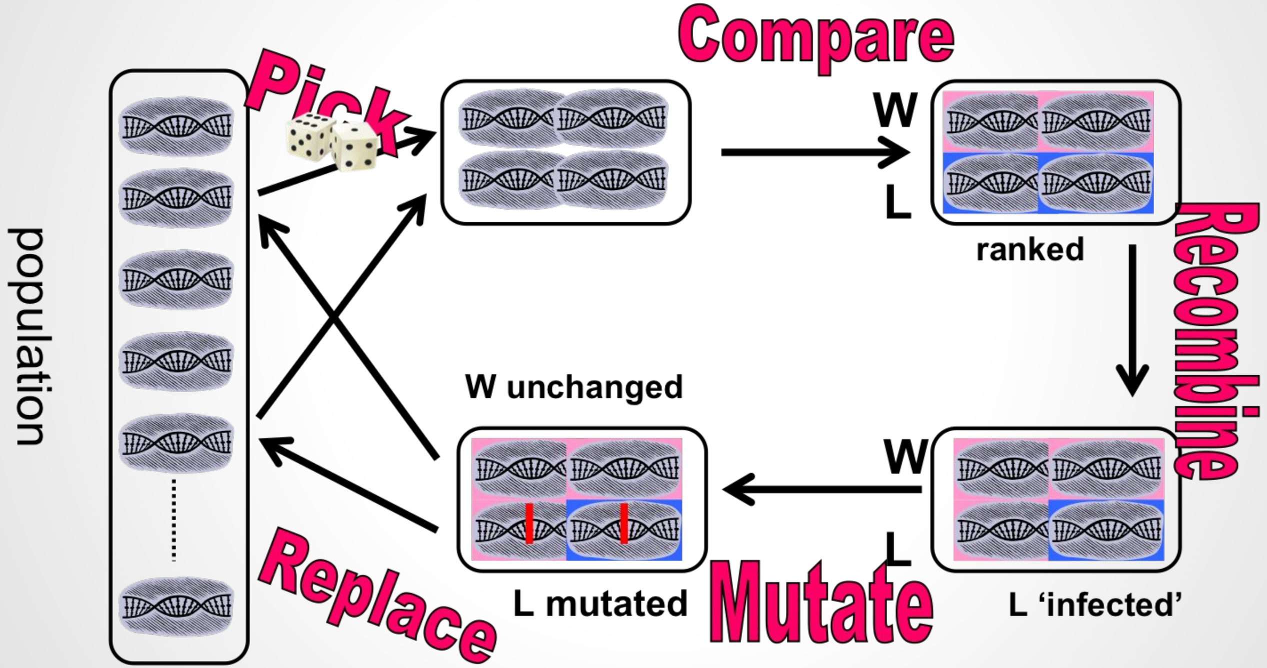
Calculate Fitness (vector, *Fit*)

for i=1:*NoTournaments* %(or for some termination condition)

- Select two individual and calculate the winner
- Copy the genes of the winner to loser with (probability, Pc)
- Mutate the Genes of loser (probability, Pm)

end

The Microbial GA (diagram)

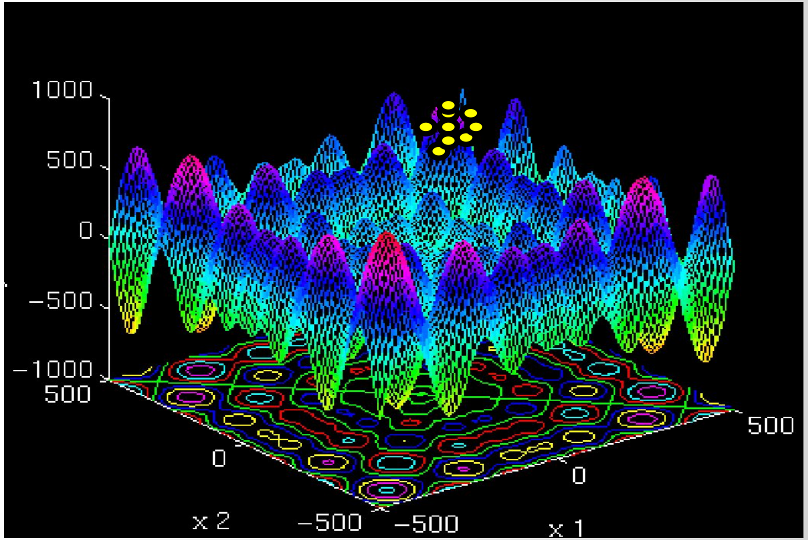


Computationally, this is quite easy …

- … because we can keep all the genotypes in a fixed array and fitness in a single vector.
- Only the Loser’s genotype is changed, within the array.
- One cycle round the loop changes one individual, n cycles is equivalent to a generation.

Diversity Maintenance

- Multiple populations may search multiple corners – but be different evolutionary runs.
- The diversity tends to collapse very quickly in GA's as the winner begins to dominate
- Maybe you can get the best of both worlds by having multiple sub-populations, with some form of limited breeding between.



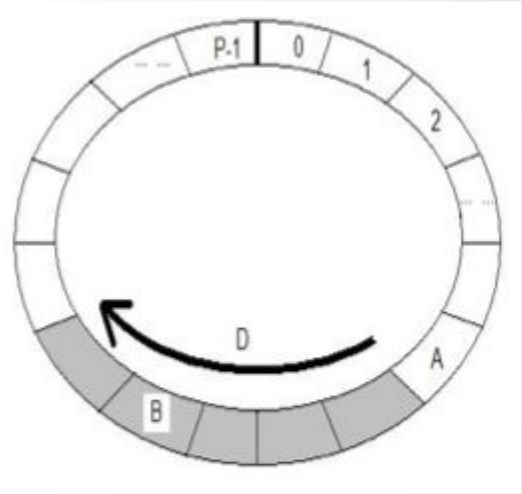
One more trick: Demes, Geographical Breeding

- One way to constrain the interbreeding is to pretend the population is distributed in space, and that they can only reproduce with other individuals near them in that space.

Genetic material diffuses

‘Trivial Geography’

- If the population is not **pan-mictic**, but instead dispersed into (overlapping) demes, we can maintain more diversity across the whole population
- GA people usually use a 2-D geography, but it looks like 1-D (a ring) is good enough.



Spector, Lee, and Jon Klein. "Trivial geography in genetic programming." *Genetic programming theory and practice III* (2006): 109-123.

Steady State with Demes EA

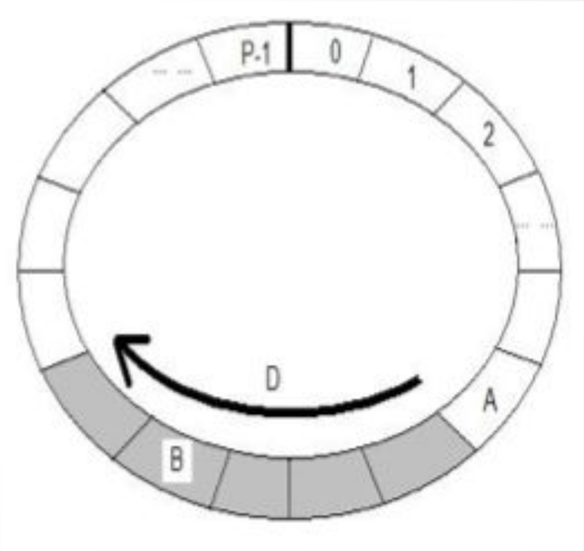
- Initialise P ‘individuals’ → Pop on a ring
- Until (satisfied or no further improvement)

  select the first competitor randomly

  - I1=rand*P+1
  - P1=Pop(I1,:)

  select next from deme size D

  - I2=(I1+1+floor(D*rand)) %NoIndividuals
  - P2=Pop(I2,:)



Combining with Demes

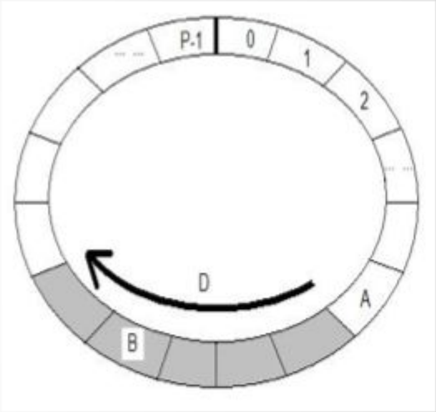
Pick + Demes

Compare

Recombine

Mutate

```
void microbial_tournament(void) {
    int A,B,W,L,i;
    A=P*rnd();                 // Choose A randomly
  B=(A+1+D*rnd())%P;   // B from Deme, %P..
    if (eval(A)>eval(B)) {W=A; L=B;}
  else {W=B; L=A;}   // W=Winner L=Loser
    for (i=0;i<N;i++) {   // walk down N genes
if (rnd()<REC)   // RECombn rate
    gene[L][i]=gene[W][i];   // Copy from Winner
if (rnd()<MUT)   // MUTation rate
    gene[L][i]^1;   // Flip a bit
    }
}
```



Elitism for free

- Many people swear by elitism.

- Elitism is the GA strategy whereby as well as producing the next generation through whichever selection, recombination, mutation methods you wish, you also force the direct unmutated copy of best-of-lastgeneration into this generation - 'never lose the best'.

Question

What difference does elitism make?

Recommendation …

- Steady state GA
- Tournament selection
- Use 1D geographical representation (demes)
- Uniform crossover
- Mutation rate very approx 1 mutation per (non-junk part of) genotype or small creep (1% of range) on real valued genotypes
- Population size usually 30 - 100 or more generally 100 * number bits of your genotype.

Next lecture

Other optimisation methods and a GA task.