Optimizing with Genetic Algorithms

by

Benjamin J. Lynch

T G
C A
A C
G T
T G
T A
G C
C T

Feb 23, 2006

Outline

- What are genetic algorithms?
  - Biological origins
  - Shortcomings of Newton-type optimizers
- How do we apply genetic algorithms?
  - Options to include
    - Encoding
    - Selection
    - Recombination
    - Mutation
  - Strategies
- What programs can we use?
  - How do we parallelize?
    - MPI, fork/wait

What are genetic algorithms? (GAs)

- A class of stochastic search strategies modeled after evolutionary mechanisms

- a popular strategy to optimize non-linear systems with a large number of variables

What are genetic algorithms? (GAs)

- A major difference between natural GAs and our GAs is that we do not need to follow the same laws observed in nature.
  - Although modeled after natural processes, we can design our own encoding of information, our own mutations, and our own selection criteria.

Definitions for today

- **Parameter** – a variable in the system of interest
- **Gene** – encoded form of a parameter being optimized
- **Chromosome** – the complete set of genes (parameters) which uniquely describe an individual
- **Locus** – the position of a piece of data within a chromosome
- **Fitness** – A value we are trying to maximize

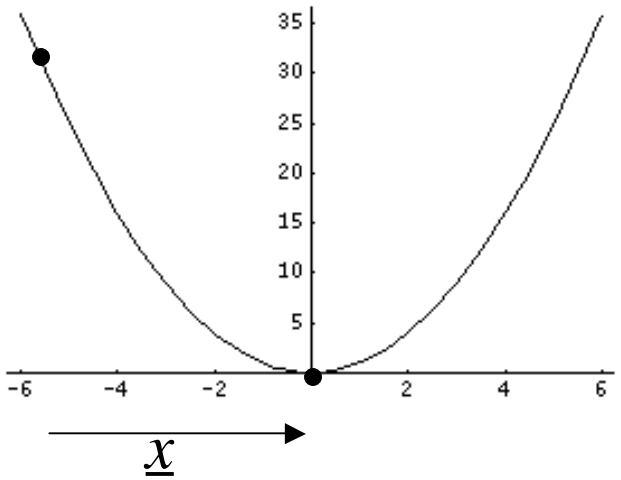
Why would we use genetic algorithms? Isn't there a simple solution we learned in Calculus?

- Newton-Raphson and it's many relatives and variants are based on the use of local information.
- The function value and the derivatives with respect to the parameters optimized are used to take a step in an appropriate direction towards a local maximum or minimum.

Newton-Raphson

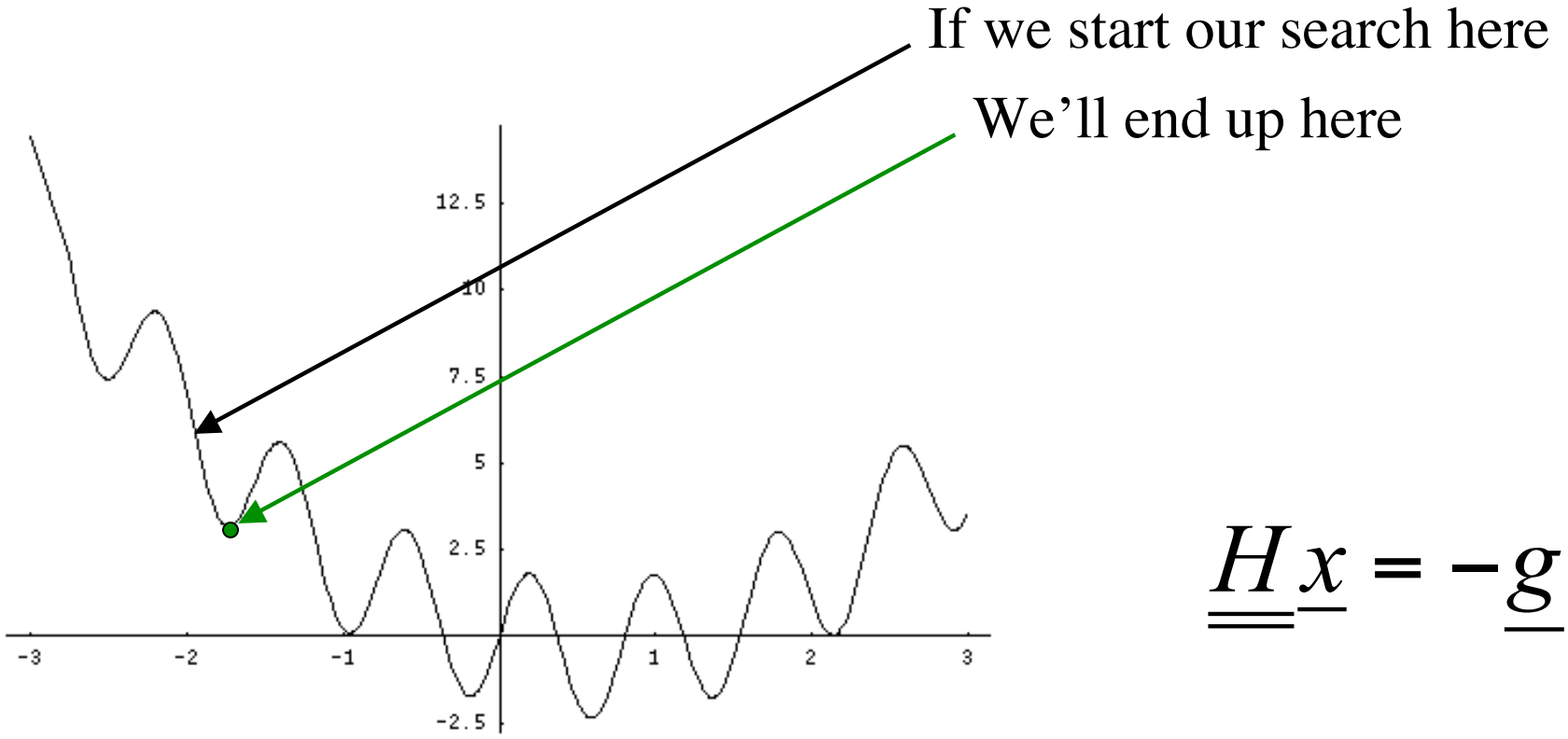
- **Perfect for a parabola!**
  - H is the Hessian (2nd derivative with respect to all parameters optimized)
  - g is the gradient
  - x is the step to minimize the gradient

$$\underline{\underline{H}}\,\underline{x} = -\underline{g}$$



Where Newton-Raphson Fails ☹

- ***A local method will only find local extrema.***



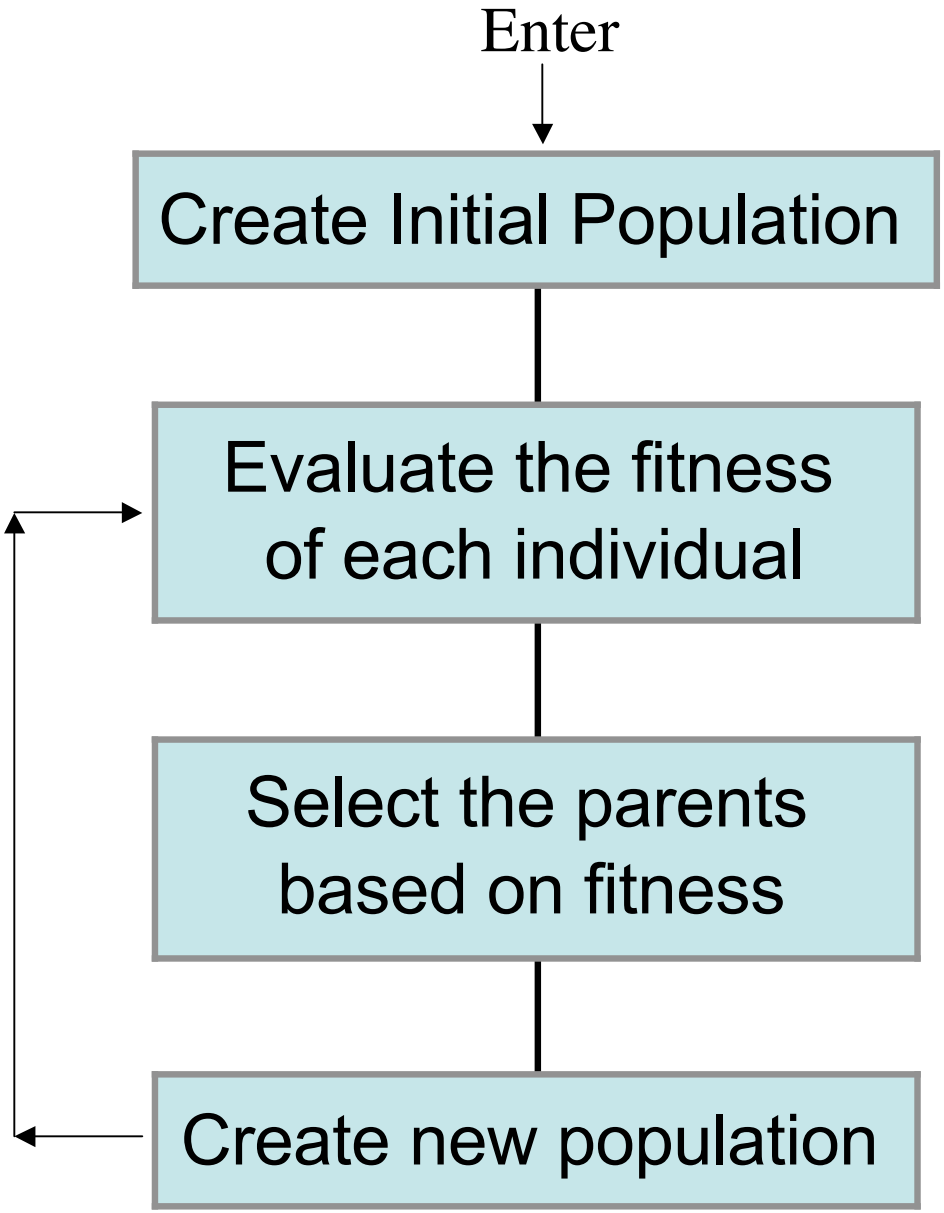
$$\underline{\underline{H}}\,\underline{x} = -\underline{g}$$

MINNESOTA SUPERCOMPUTING INSTITUTE

How do we use GAs to optimize the parameters we're interested in?

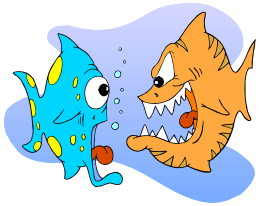
- Choose parameters to optimize
- Determine chromosomal representation of parameters
- Generate initial population of individuals (chromosomes)
- Evaluate fitness of each individual to reproduce
- Allow selection rules and random behavior to select next population

Genetic Algorithm



Evaluation

Selection

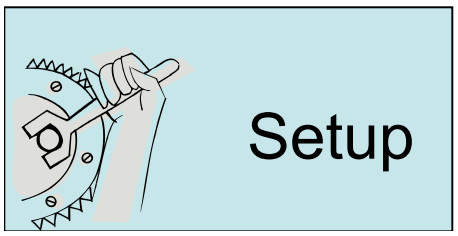


Recombination

MINNESOTA SUPERCOMPUTING INSTITUTE

Setting up the problem

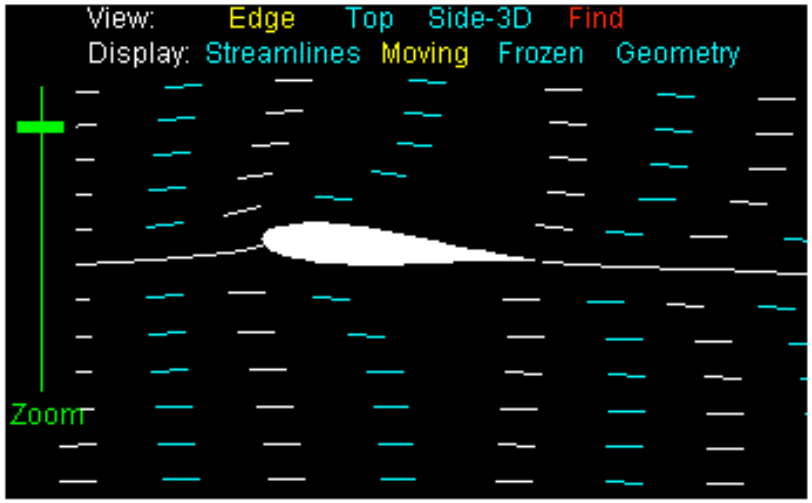
- **This is the most difficult step!**
- Choose the parameters you want to optimize

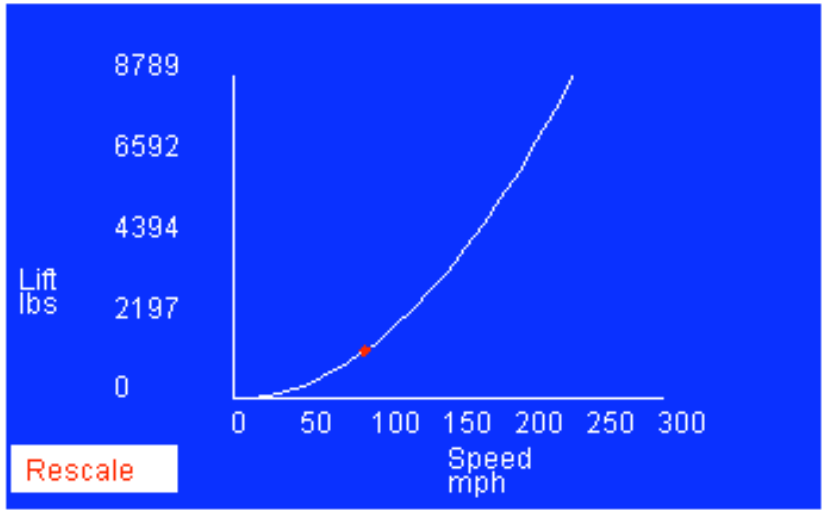


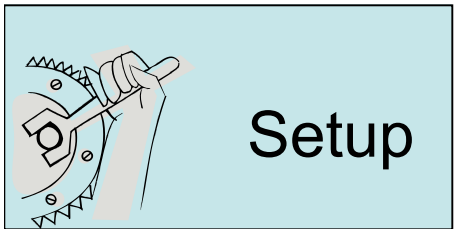
Determine the Parameters

Airfoil Example:

- Constant wing area
- Variable camber
- Variable chord at root
- Variable chord at tip
- Span (function of chords and wing area)



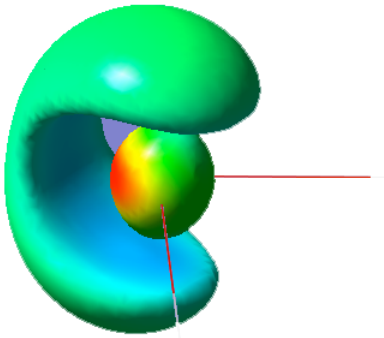


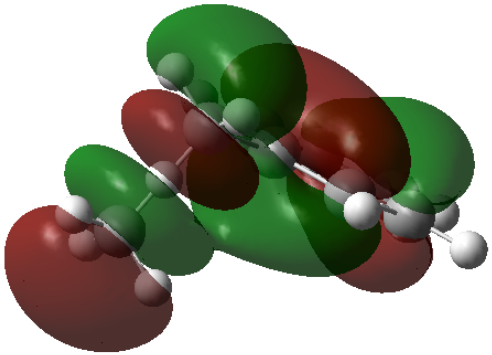


MINNESOTA SUPERCOMPUTING INSTITUTE

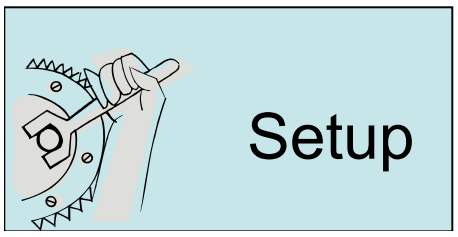
Density Functional Example:

- Choose parameters to be all the variables in the gradient-corrected exchange terms.



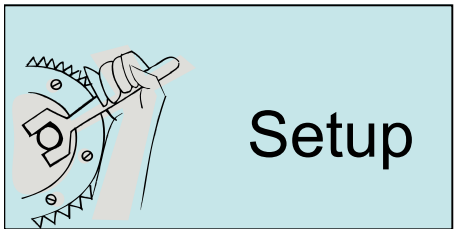


$$E_X^{GGA} = -\sum_\sigma \int \left(\frac{3}{2}\left(\frac{3}{4\pi}\right)^{1/3} + \frac{ax^2 - bx^2 e^{-cx^2} - dx^f}{1 + 6ax\sinh^{-1} x - \frac{dx^f}{A_x}} \right) \rho_\sigma(r)^{4/3}\, dr$$



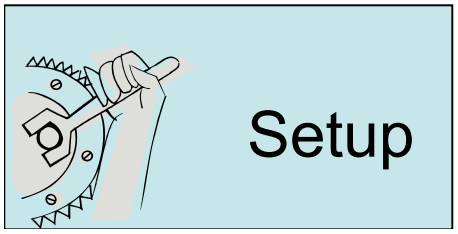
Evaluate Your Fitness

- For every problem, there is something you want to maximize or minimize.
- The standard convention is to maximize a function with a GA.



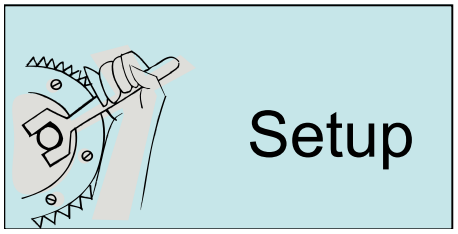
Fitness

- We need to evaluate fitness of each individual.
- The fitness will be used to bias the next generation towards better genes.



Fitness

- We must first define what fitness is! You must come up with a single metric that will be used to compare 2 possible solutions and decide which is better.

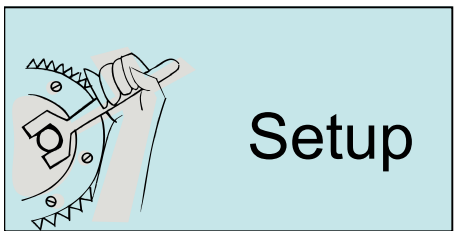


- For an airfoil, this might be a function of drag and lift

$$Fitness = w_1 Lift(P) - w_2 Drag(P)$$

- It may depend on a set of simulations at different speeds, different angles of attack, etc.

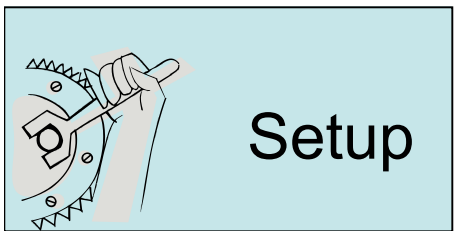
$$Fitness = \sum_{i}^{simulations} w_i Lift(P) - \sum_{i}^{simulations} w_i Drag(P)$$



- For an empirical density functional, the fitness might be a weighted RMS deviation from experimental values.

$$Fitness = -\sum_{i}^{dataset} (E_i^{Calc}(P) - E_i^{Exp})^2$$

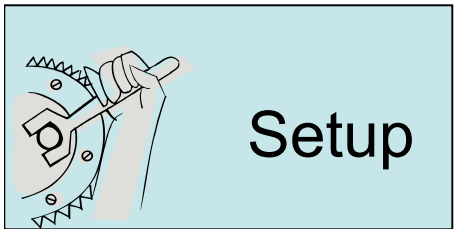
Note: we toss in a "–" because we always Want to maximize the fitness.



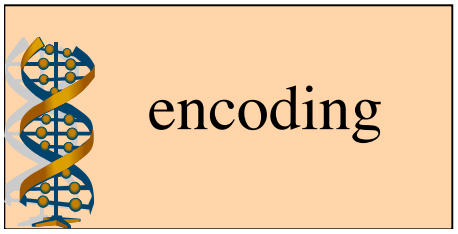
- Check that your problem is well-suited for optimization with a GA.

- Your fitness function will need to be evaluated thousands of times. Make sure you have the resources.

- If a GA is too expensive, you still might be able to simplify your problem and use a GA to find regions in the parameter space of interest.



- Determine chromosomal representation of parameters
- Parameters can be encoded in binary, base-4 base-10, etc.



- After you decide how to encode the parameters, you must decide on the domain of your parameters. This is entirely dependent on your problem. You will want to allow your parameters to be anything physically reasonable (if you're solving a physical problem)

- Create an initial population with randomized chromosomes

encoding

Create Initial Population

- Population size is chosen (1-10 individuals/parameter optimized for most applications)
- Parameters to be optimized are encoded.
  - Binary, Base 10
  - Let's say we have 2 parameters with initial values of 32 and 13. In binary and base 10 they would look like:

Chromosome of the individual

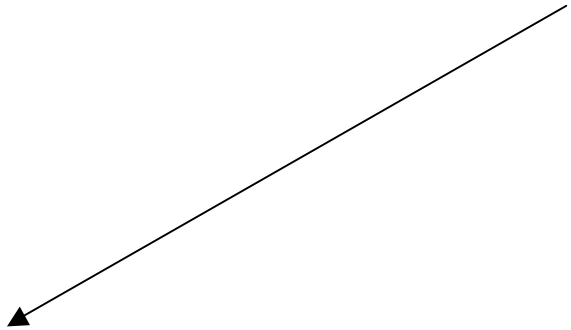
3213

encoding

How binary genes translate into parameters

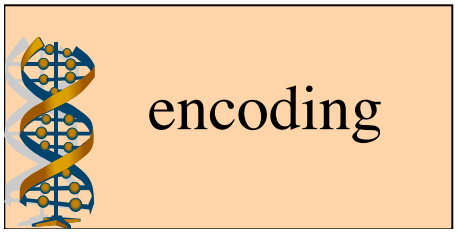
1010111010010101111010000100110

698 349 38



$$P = \frac{38}{2^{10} - 1}(P_{\max} - P_{\min}) + P_{\min}$$

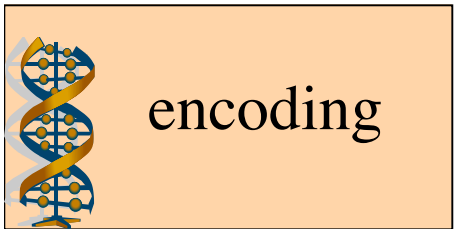
You need to understand the system you are optimizing in order to determine the proper parameter range



MINNESOTA SUPERCOMPUTING INSTITUTE

Create Initial Population

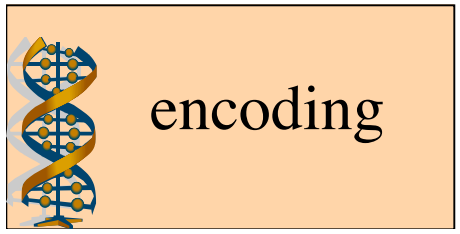
- After we choose a population size and encoding method, we must choose a maximum range for each parameter. Ranges for parameters should be determined based on what would be physically reasonable (if you're interested in solving a physical problem).



- Generate initial population of individuals (chromosomes)

- The initial population can be generated by randomizing the genes for each chromosome of the initial population

- You can set the parameters for a few individuals if you want. This might speed up the process.

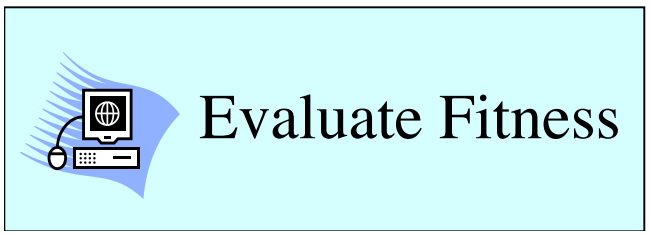


Review Steps in a GA

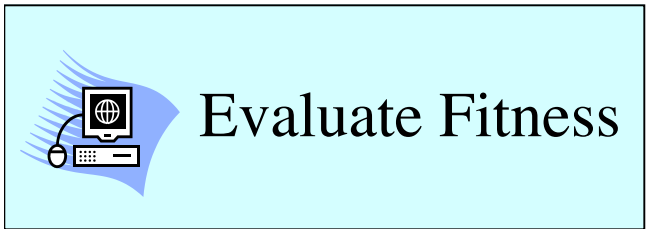
1. Initialization of population 
2. Evaluation of fitness
3. Selection
4. Recombination
5. Repeat 2-4

Evaluation

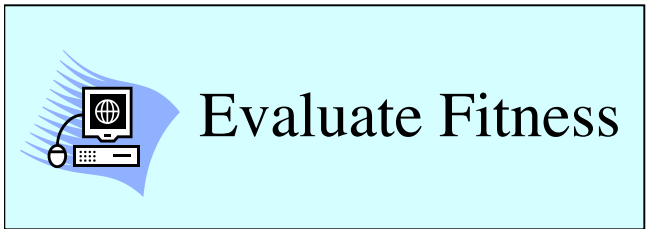
- Assign a fitness value to each individual based on the parameters derived from its chromosome



- The fitness function is somehow based on the genes of the individual and should reflect how good a given set of parameters is.
  - Lift-drag , low drag airfoil
  - Ability of a density functional to better predict chemical phenomena
  - Swimming speed of a robotic fish
  - Power output of a chemical laser



- Evaluation of the fitness is the computationally-intensive portion of a GA optimization
- Each chromosome holds the information that uniquely describes an individual.
- Each chromosome/(parameters set)/individual can be evaluated separate from the other individuals.
  - GA optimizations are typically described as embarrassingly parallelizable
- The evaluation of the chromosomes reduces down to a fitness value for each individual which will be used in the next step



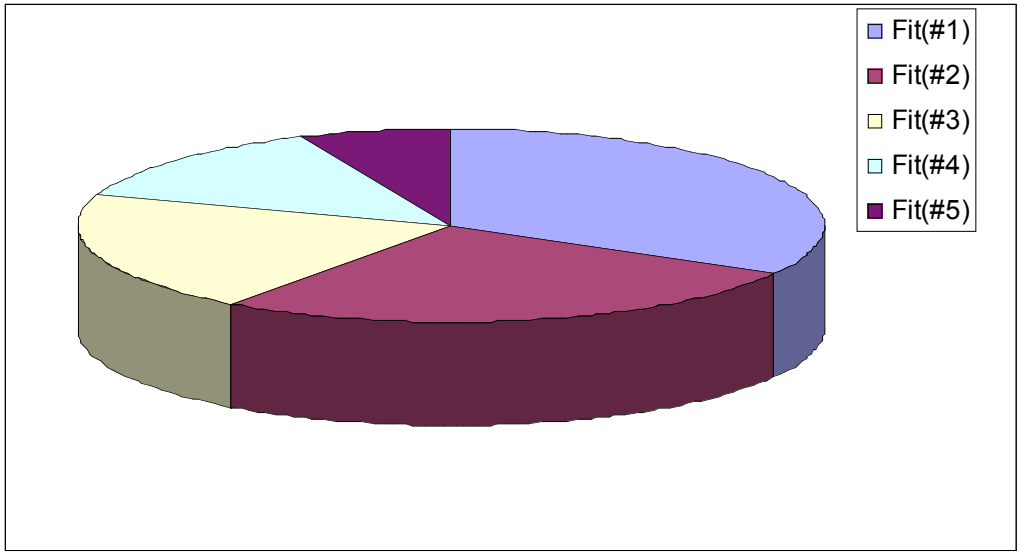
Selection

- Allow selection rules and random behavior to select next population

- The parents must be selected based on their fitness.
- The individuals with a higher fitness **must** have a higher probability of having offspring.
- There are several methods for selection.

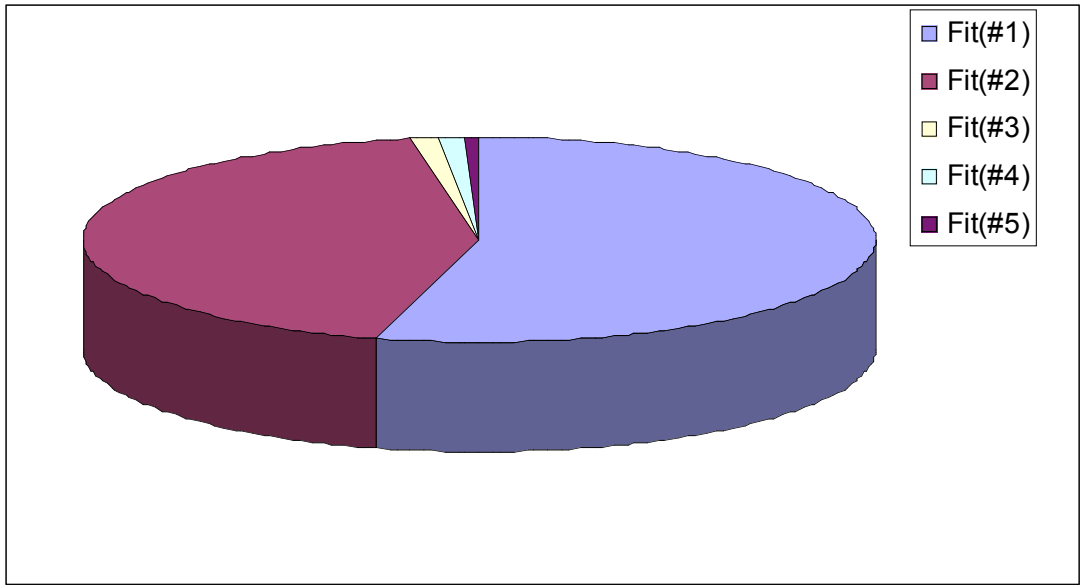
Roulette Wheel Selection

- Probability of parenthood is proportional to fitness.
- The wheel is spun until two parents are selected.
- The two parents create one offspring.
- The process is repeated to create a new population for the next generation.



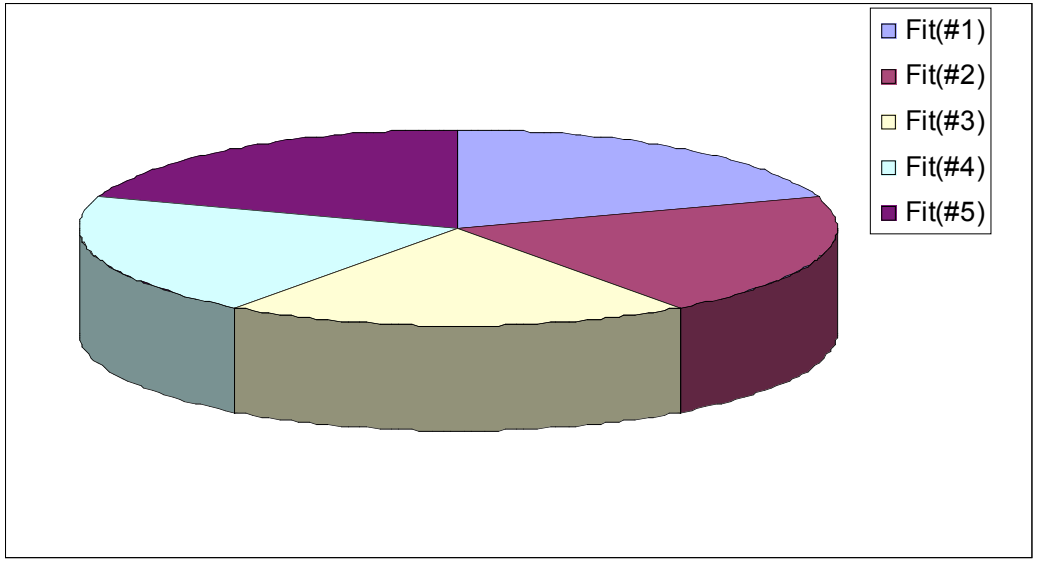
MINNESOTA SUPERCOMPUTING INSTITUTE

- Roulette wheel selection has problems if the fitness changes by orders of magnitude.
- **If two individuals have a much higher fitness, they could be the parents for every child in the next generation.**



Another Reason Not to Use the Roulette Wheel

- If the fitness value for all individuals is very close, the parents will be chosen with equal probability, and the function will cease to optimize.
- Roulette selection is very sensitive to the form of the fitness function and generally requires modifications to work at all.

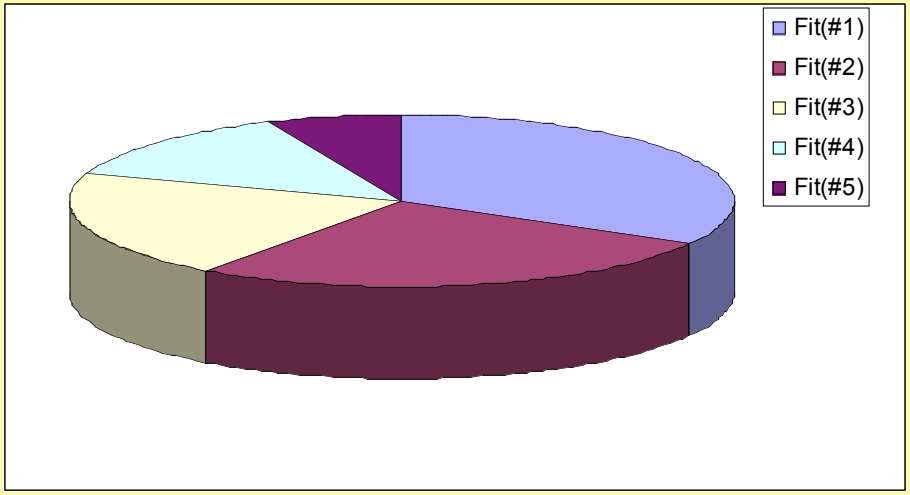


MINNESOTA SUPERCOMPUTING INSTITUTE

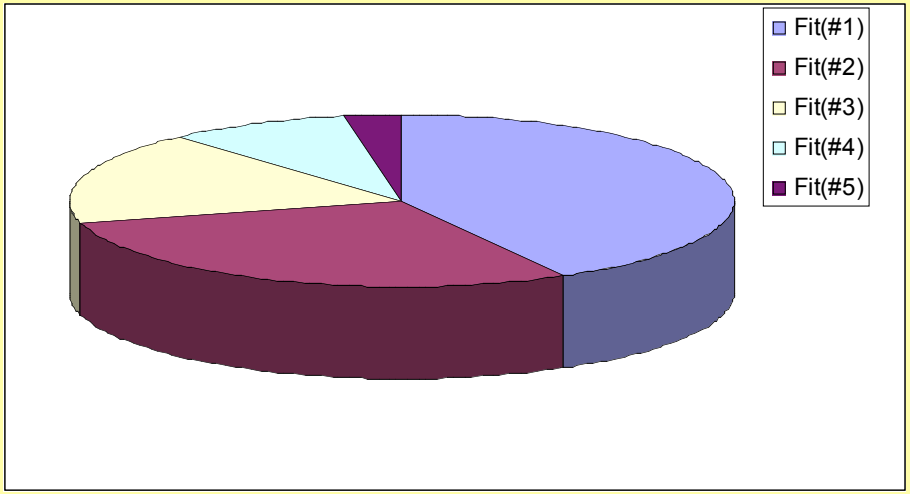
Rank Selection

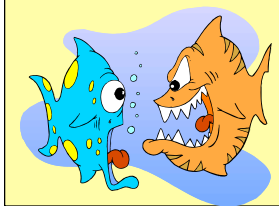
- All individuals in the population are ranked according to fitness
- Each individual is assigned a weight inversely proportional to the rank (or other similar scheme).

$$\frac{1}{rank}$$



$$\frac{1}{(rank)^{1.7}}$$

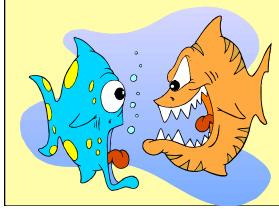




Selection

Tournament Selection

- 4 individuals (A,B,C,D) are randomly selected from the population. Two are eliminated and two become the parents of a child in the next generation

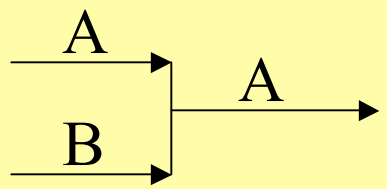


Selection

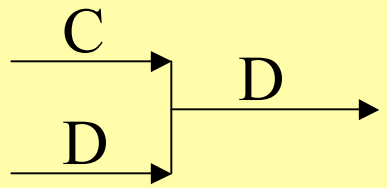
Tournament Selection

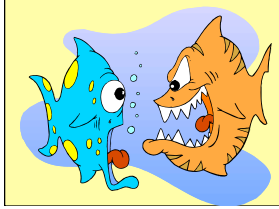
- Selection of parents continues until a new population is completed.
- Individuals might be the parent to several children, or no children.

Fitness(A) > Fitness(B)



Fitness(D) > Fitness(C)





Selection

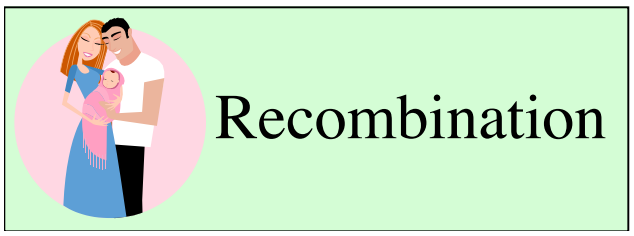
Similarities Between Tournament and Rank Selection

- Tournament selection is very similar to rank selection in the limit of a large population when we assign a weight of 1/rank.

Fraction of population	Fitness
$\frac{9}{16}$	Both parents were above the median
$\frac{6}{16}$	One parent was above the median
$\frac{1}{16}$	Neither parent was above the median

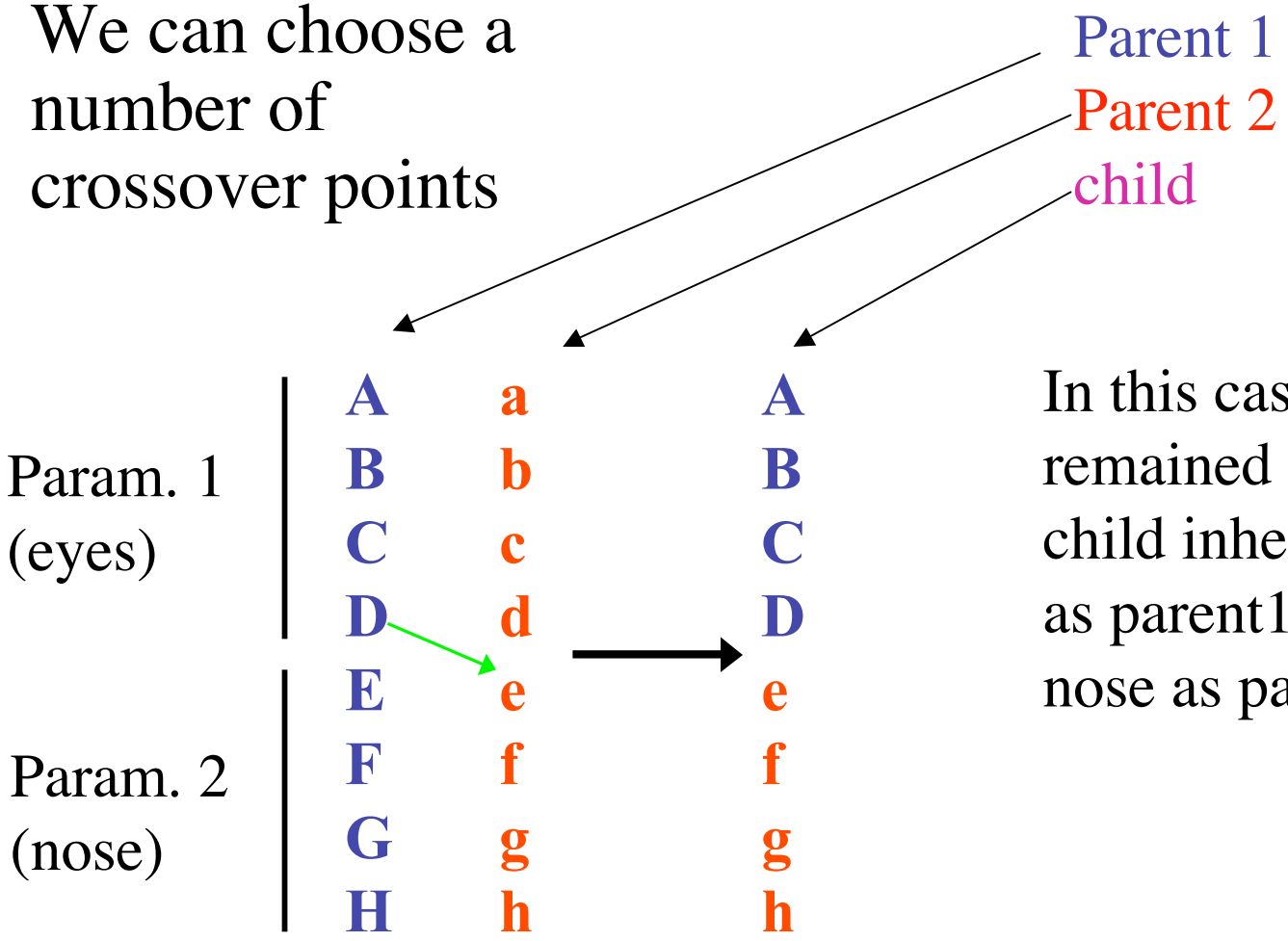
Recombination

- Using the chromosomes of the parents, we create the chromosome of the child

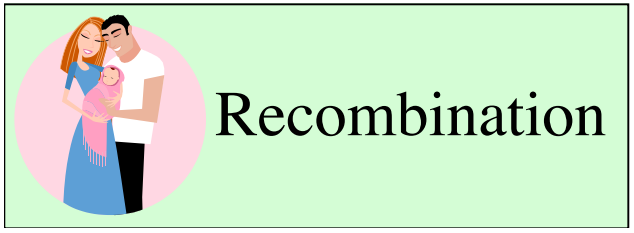


Recombination with crossover points

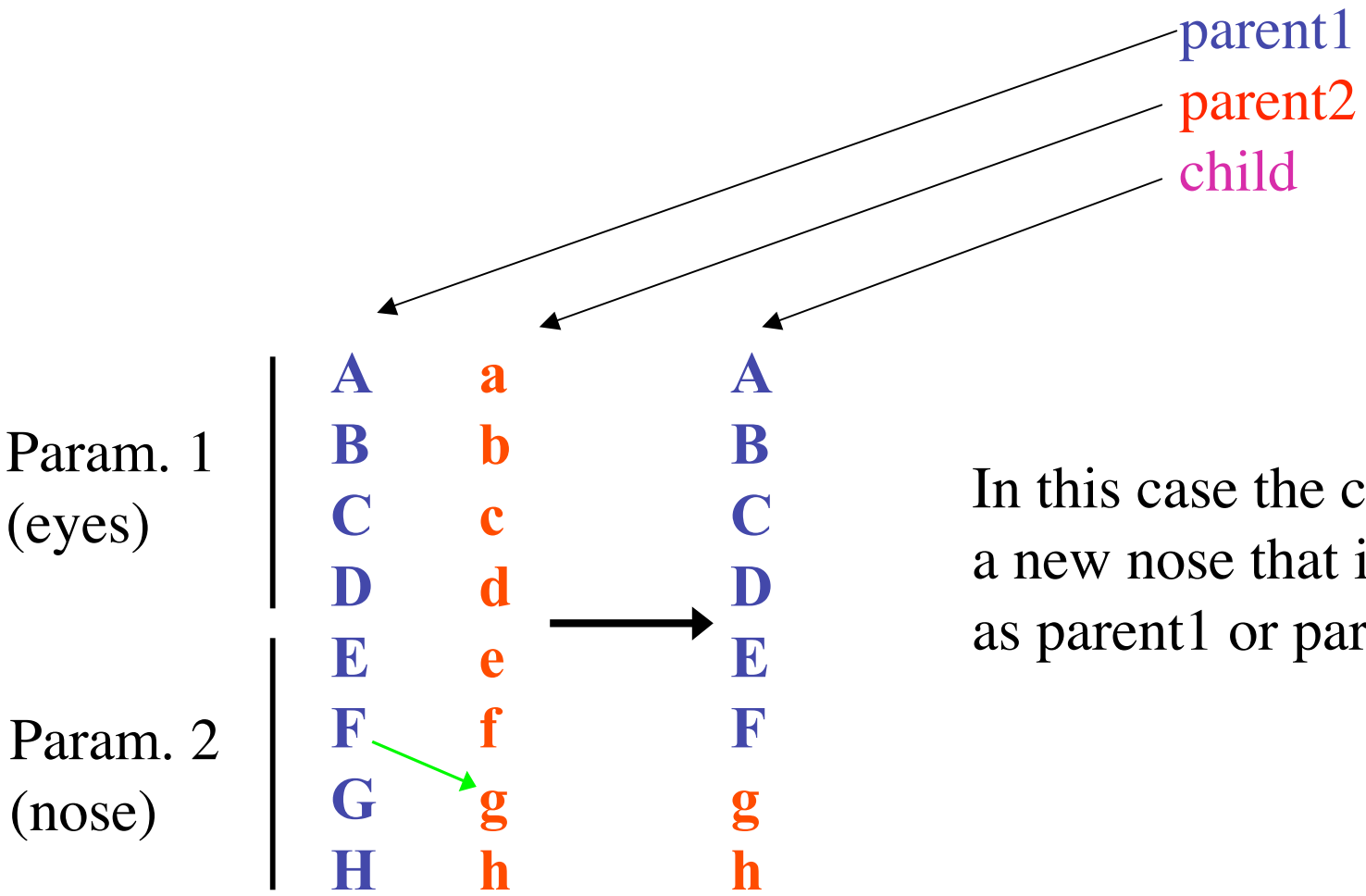
- We can choose a number of crossover points



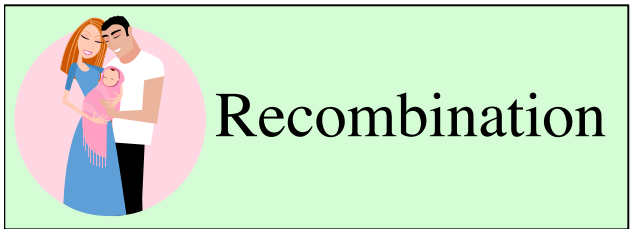
In this case, the parameters remained intact, and the child inherited the same eyes as parent1 and the same nose as parent2.



crossover point occurs within a parameter

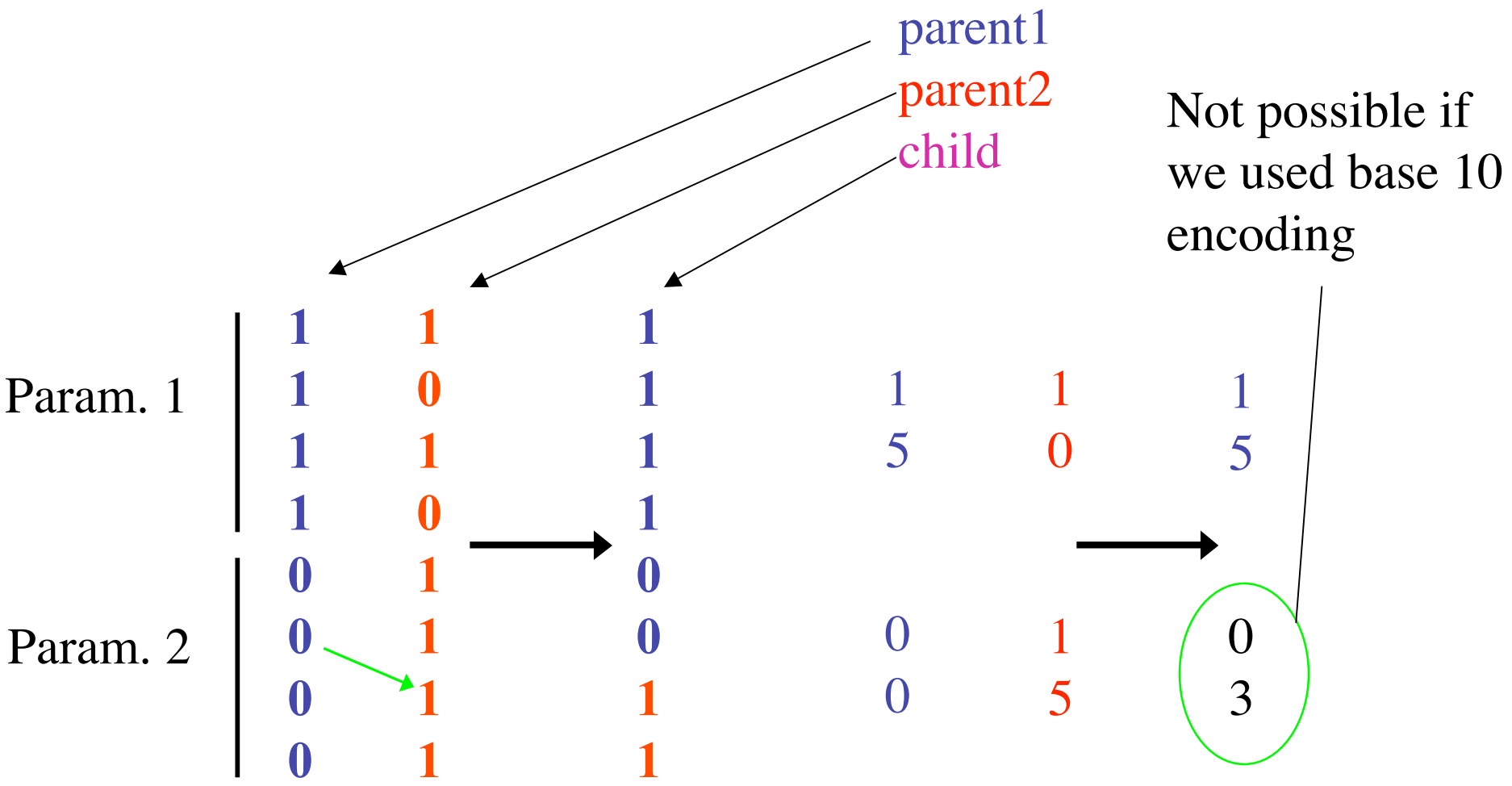


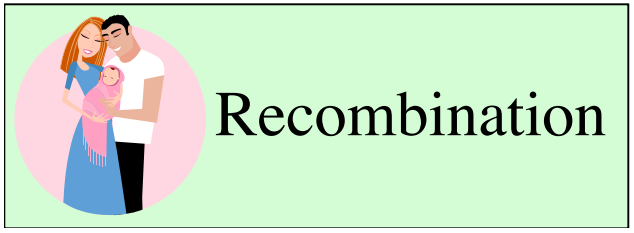
In this case the child will have a new nose that is not the same as parent1 or parent2.



MINNESOTA SUPERCOMPUTING INSTITUTE

representation of parameters becomes important.





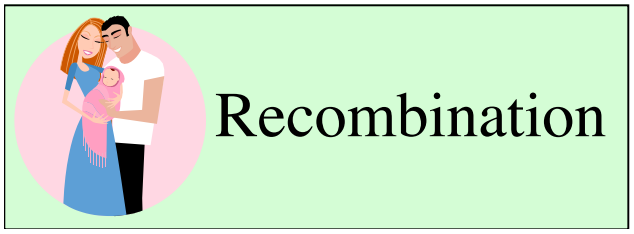
MINNESOTA SUPERCOMPUTING INSTITUTE

Recombination – Uniform Crossover

- Uniform crossover
  - No limit to crossover points

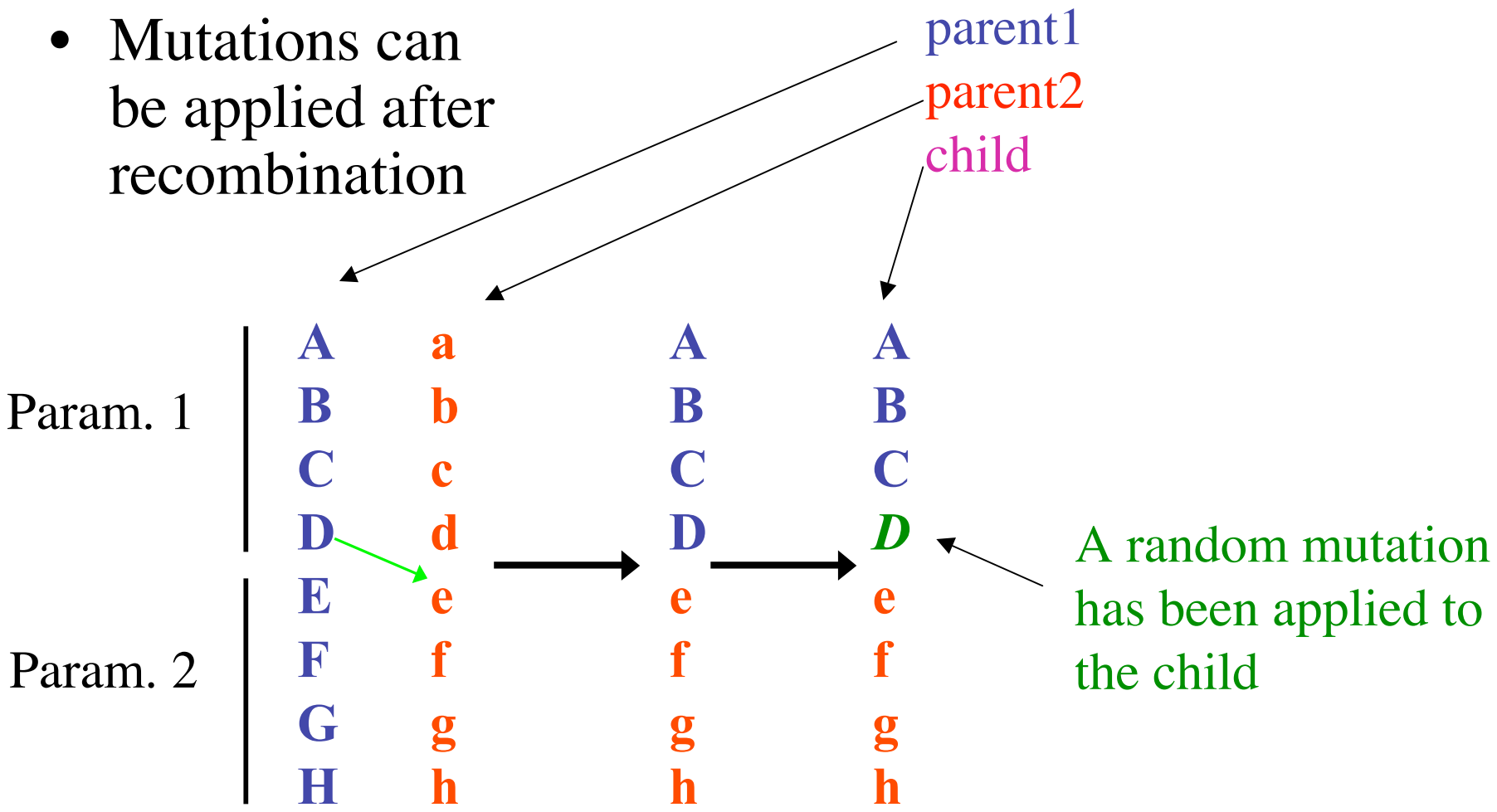
A	a	a
B	b	B
C	c	c
D	d	D
E	e	e
F	f	f
G	g	g
H	h	H

Allows more variation in offspring and decreases need for random mutations

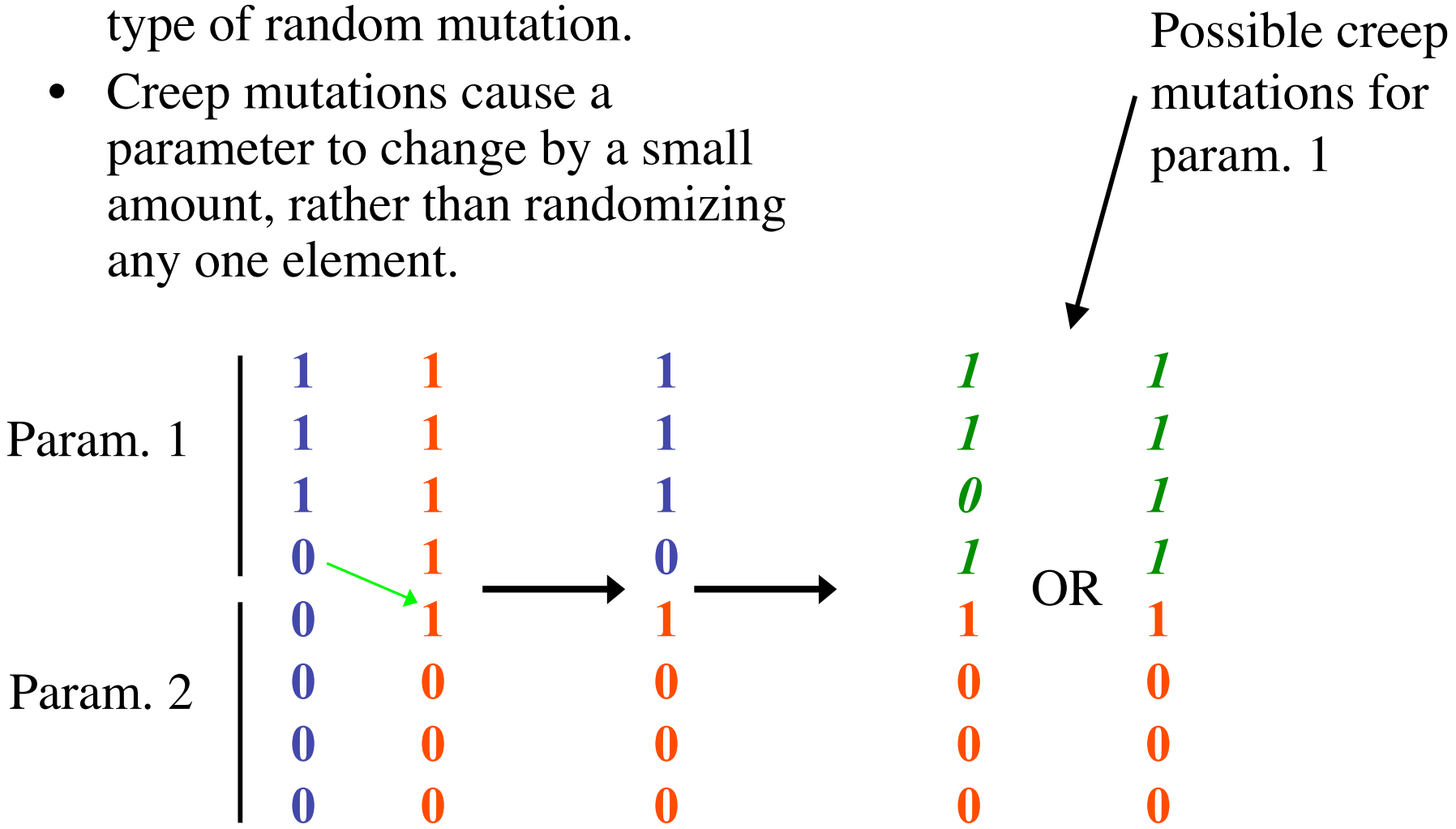


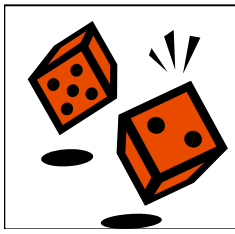
MINNESOTA SUPERCOMPUTING INSTITUTE

- Mutations can be applied after recombination



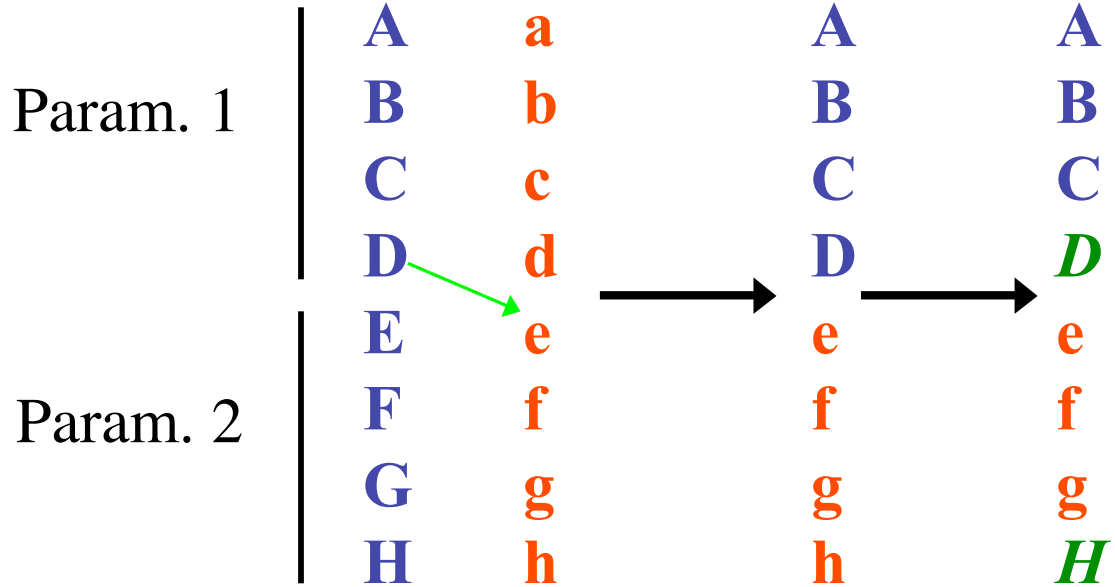
- Creep mutations are a special type of random mutation.
- Creep mutations cause a parameter to change by a small amount, rather than randomizing any one element.





Random mutations

- The desirable frequency of mutations depends greatly on the other GA options chosen.

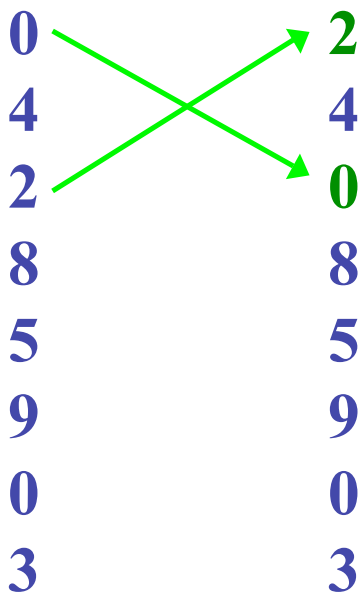


MINNESOTA SUPERCOMPUTING INSTITUTE

Other Operators for Recombination

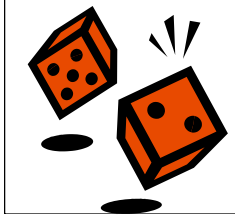
- Other rearrangements of information are possible

- Swap locus



- Swap entire genes

0	5
4	9
2	0
8	3
5	0
9	4
0	2
3	8



Random mutations

Elitism

- Elitism refers to the safeguarding of the chromosome of the most fit individual in a given generation.
- If elitism is used, only *N*-1 individuals are produced by recombining the information from parents. The last individual is a copy of the most fit individual from the previous generation.
- This ensures that the best chromosome is never lost in the optimization process due to random events.

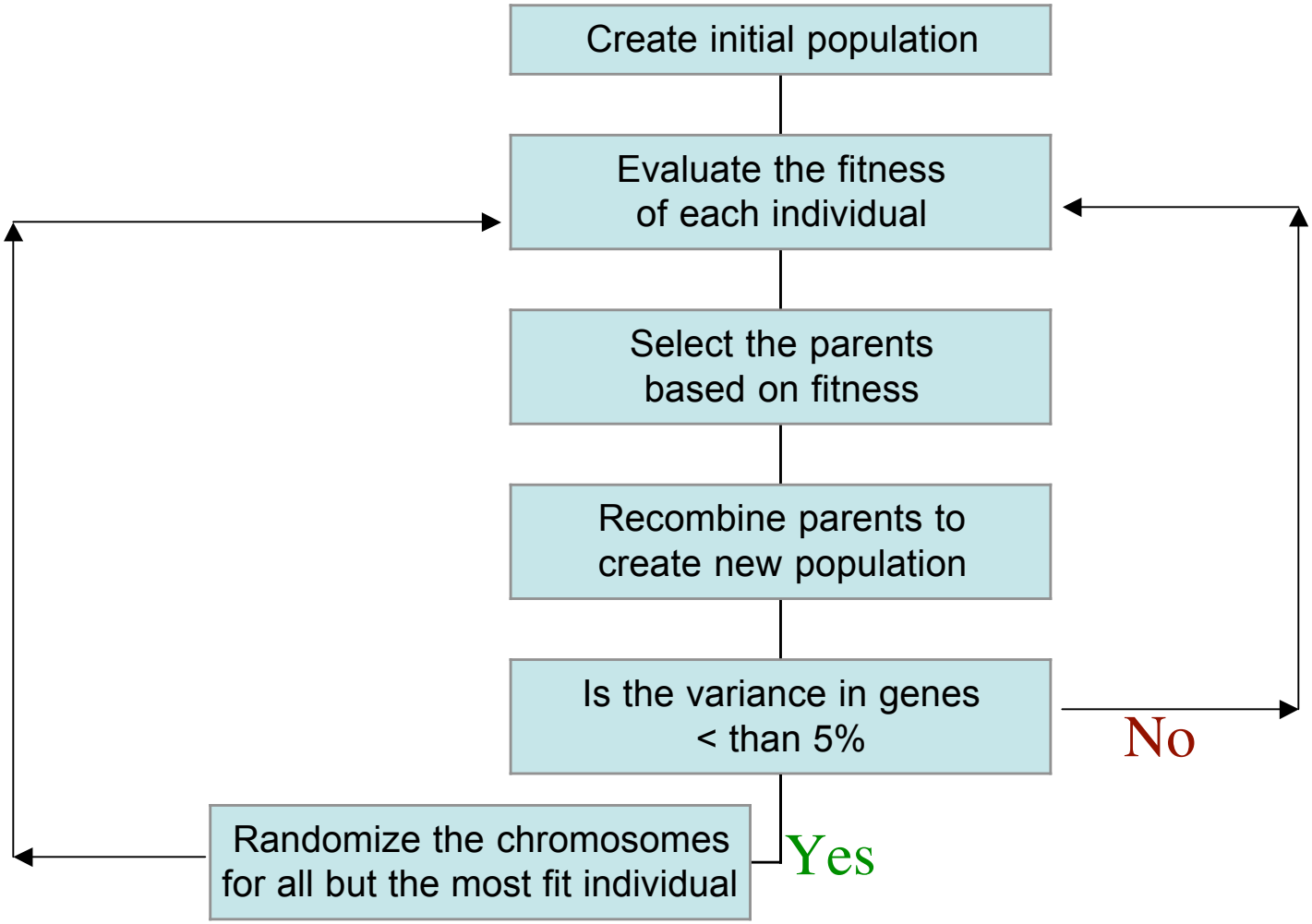
More GA Options

- Separate “islands” with populations that interact infrequently.
- Use “male” & “female” populations
- “Alpha male” selection
- Use 3 parents for each offspring
- Use 3 sexes
- Recessive genes
- …. and many more. Most of which are only useful for very specific types of fitness functions.

Micro-GA

- Small population size of 1 individual per parameter optimized.
- No random mutations.
- When the genetic variance is below a certain threshold (~5%), the most fit individual goes on, while the chromosomes of all other individuals are randomized
- Cycle this process

Overview of a micro-GA



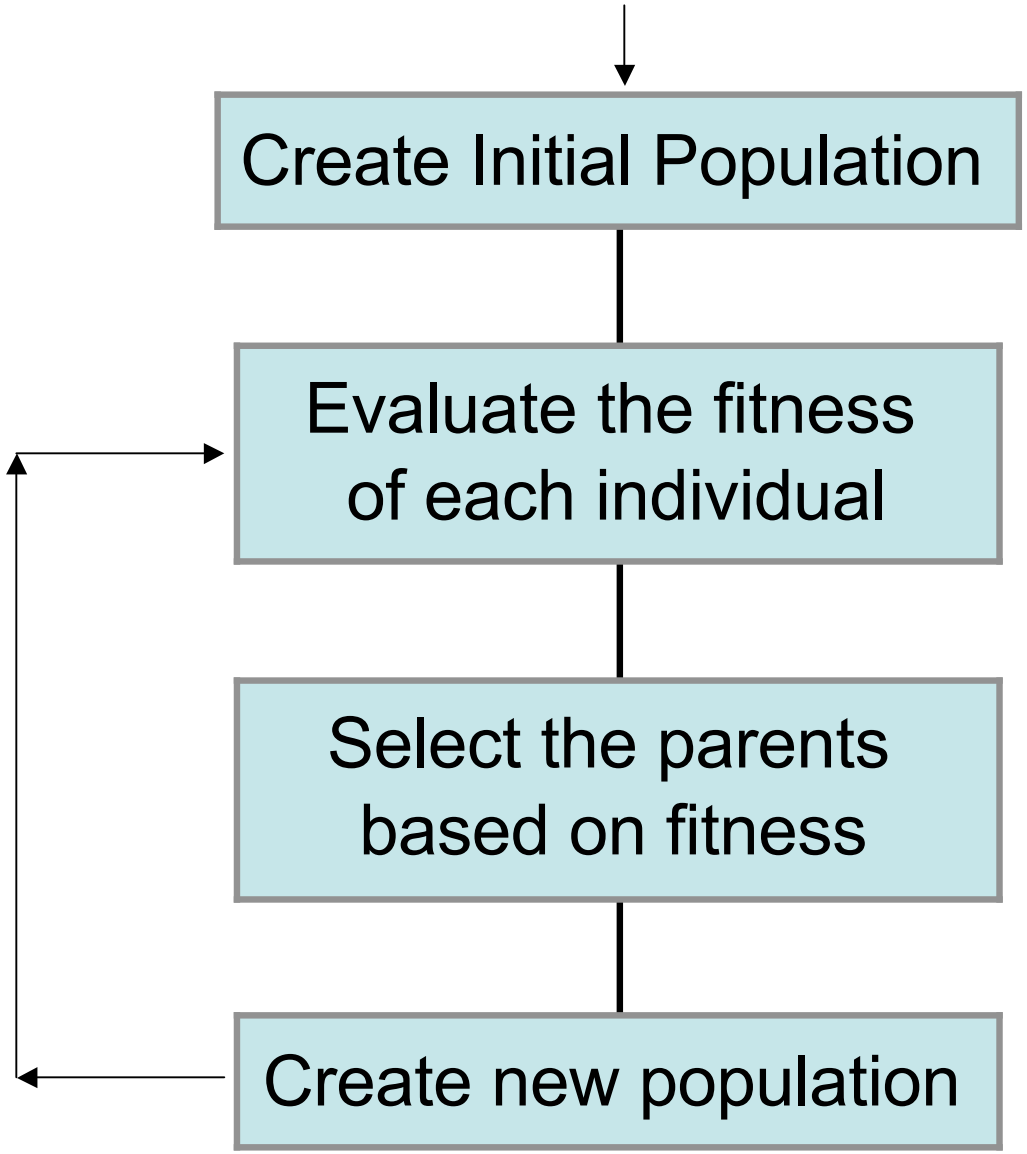
MINNESOTA SUPERCOMPUTING INSTITUTE

Micro-GA Pros and Cons

- **Tends to be very efficient in terms of total CPU time.**
- **Robust algorithm with no need to tinker with random mutation parameters**

- **It uses a smaller population, and therefore has less potential for parallelization.**

Review



Evaluation of the fitness is the time-consuming portion

Evaluation

Selection

Recombination

MINNESOTA SUPERCOMPUTING INSTITUTE

How do you know if you've found the absolute maximum?

- **You don't**
- Even GAs are not a "black-box" optimizer for any function
- You can gain confidence by running several optimizations with different starting parameters, different algorithm options, and different parameter ranges.

Which GA options are good picks for my system?

- **Start with robust algorithms**
  - Micro-GA
  - Binary encoding
  - Tournament selection
  - Uniform crossover at gene boundaries

- **If you are unsatisfied with the progress you can change a couple options**
  - Allow crossover everywhere to introduce more variety between each generation.
    - Change to base-10 encoding as well, so it isn't too random
  - Use rank selection with a weight of $1/\text{rank}^3$ to heavily favor the best individuals.

How might one parallelize a GA?

- The GA calculations are minimal. An optimization might require 1000 generations, and each generation is dominated by the cost to evaluate the fitness.
- Standard serial GA programs can handle the GA routines with ~1 ms of cpu time, while your fitness routine can parallelize the fitness evaluations.

How might one parallelize a GA?

- Evaluating the fitness of an individuals is independent of the rest of the population.
  - You can easily run your GA on *N* processors where *N* is your population size.

- Each individual can often be run in parallel as well
  - This depends on the program you are using to evaluate the fitness.

Method to Parallelize a GA

- MPI is always a good choice if you're already familiar the language. This option would also enable GA algorithms with "islands" on heterogeneous clusters.

- Fork/wait would be very easy
  - Can be done in several languages

```
#!/usr/bin/perl
use Parallel::ForkManager;
$max_process=4;

$pm = new Parallel::ForkManager($max_process);
@all_chromosomes=(@ARGV)

# enter main loop, no more than $max_process
# children will run at a time
foreach $chromosome (@all_chromosomes) {
    my $pid = $pm->start and next;
# prepare an appropriate input file with
# this set of parameters
    &writeinput($chromosome);
# run the program to evaluate the fitness
    &evalfitness($chromosome);
    $pm->finish;
}

# make sure that all the processes are done
$pm->wait_all_children;

#parse output and return to our GA
&parse_output;
```

Fork method to parallelize with Perl on a shared-memory machine

MINNESOTA SUPERCOMPUTING INSTITUTE

GA Programs Available

- David Carrol's GA Driver
  - http://cuaerospace.com/carroll/ga.html
- GAUL
  - http://gaul.sourceforge.net/
- GALOPPS (already parallelized)
  - http://garage.cps.msu.edu/software/galopps/index.html
- Simple Generalized GA
  - http://www.skamphausen.de/software/AI/ga.html

Other GA Resources Available

- Me
  - blynch@msi.umn.edu
  - 612-624-4122

Questions?

blynch@msi.umn.edu 4-4122

help@msi.umn.edu 6-0802

MINNESOTA SUPERCOMPUTING INSTITUTE