# *Using genetic algorithms to solve complex problems*

*Julius van der Werf*

*School of Rural Science and Agriculture*
*University of New England*
*Armidale, Australia*

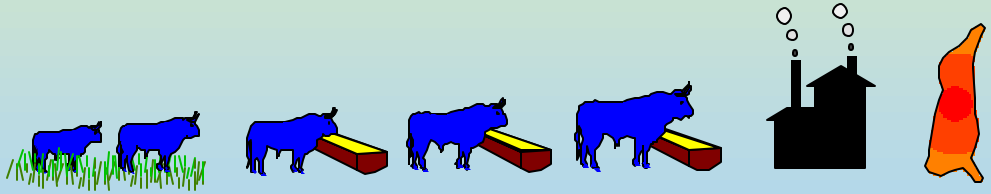
The University of New England AUSTRALIA

# Complex Problems

- Multi-dimensional
- Multi-objective
- Multiple local optima
- Flat surfaces
- Non-differentiable
- Variables are highly interactive
- Messy

# complex (animal) production problems

- Optimal resource allocation
  - Feeding strategies
  - Logistics of supply



  - Total Resource Management: **TRM**

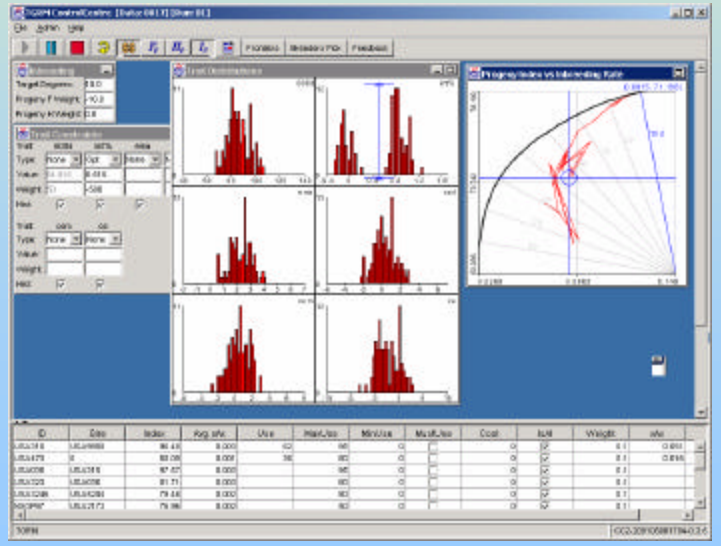
# complex animal breeding problems

- Breeding program design
  - Mate selection
  - Development of synthetics / genetic resources

**TGRM**

- Breeding objectives
  - Non-linear profit space
  - Selection index weights (incl. QTL)

- Modeling
  - multiple QTL models
  - Growth models
  - Haplotype configuration



# Solving complex problems

- Non-random numerical techniques (deterministic search algorithms)
  - Linear programming
  - Dynamic Programming
  - Optimal Control Theory
  - Simplex algorithm

tedious

# Solving complex problems

- All deterministic search algorithms
  - Need to break the ‘curse of multidimensionality’
- Exhaustive (finite) search algorithms
  - Job increases exponentially with problem size

# Random search or stochastic algorithms

- Efficient
- Optimal solution not guaranteed
  - But……..

Perform better than other algorithms in complex problems → Real Time solutions

# Random search or stochastic algorithms

- Simulated Annealing
- Tabu Search
- Evolutionary Algorithms
  - Genetic Algorithm
  - Differential Evolution
  - Memetic Algorithms (from selfish gene: meme)
- Immune Systems
- Ant Colony Optimization

# Genetic Algorithms

- GA = “Get away with Algebra”
- 30 years old

- Fitness = f(x)
  - x = vector with parameters

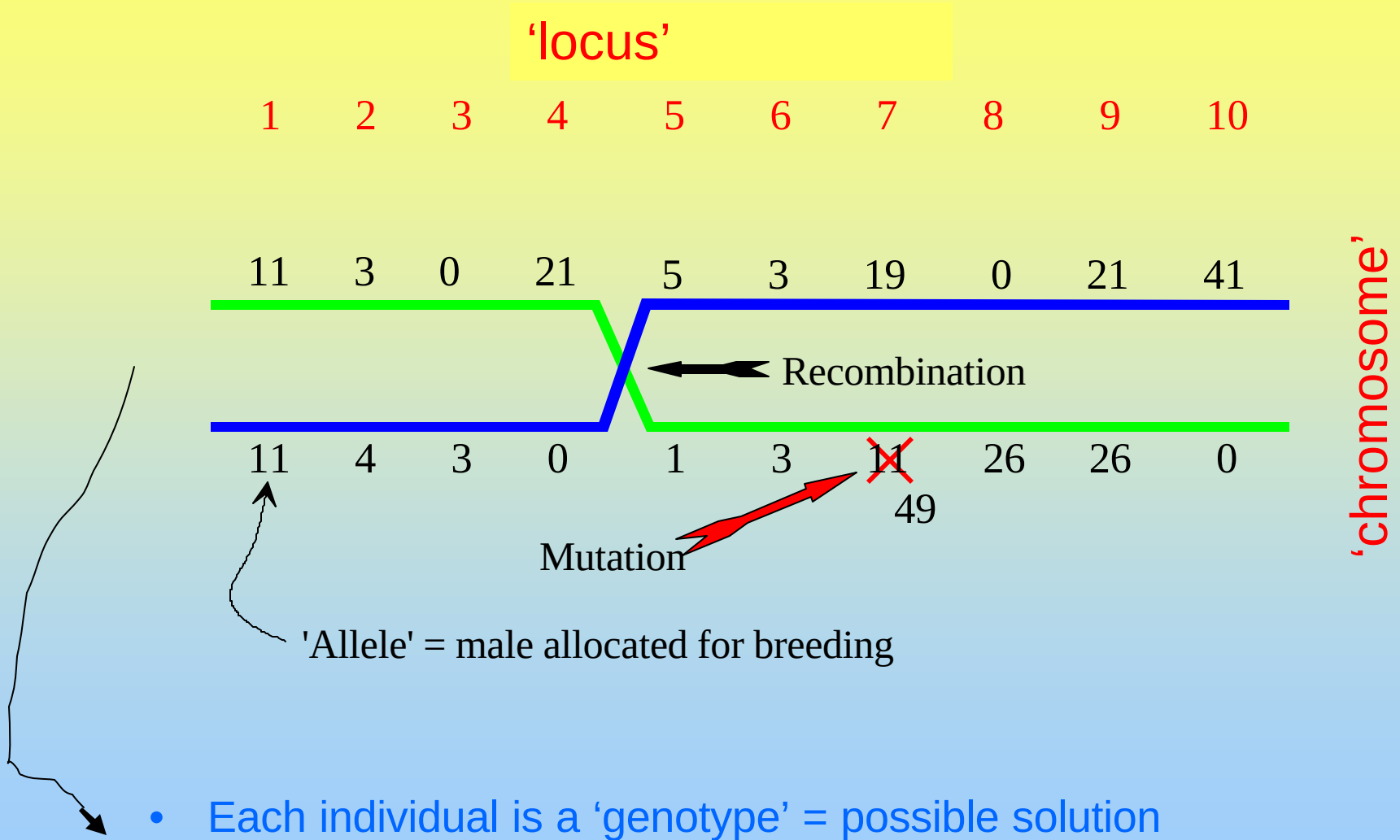
  - Repeated cycles of
    - mutation
    - recombination
    - selection
    - mating of parents

# The Theory of Evolution

- Fitness selection
- Mutation
  - To create variation
  - Need just enough, but limited
  - Can it work by itself?
    - The value of sex
- Recombination helps
- Selection……etc

As geneticists we know how powerful the concept is!

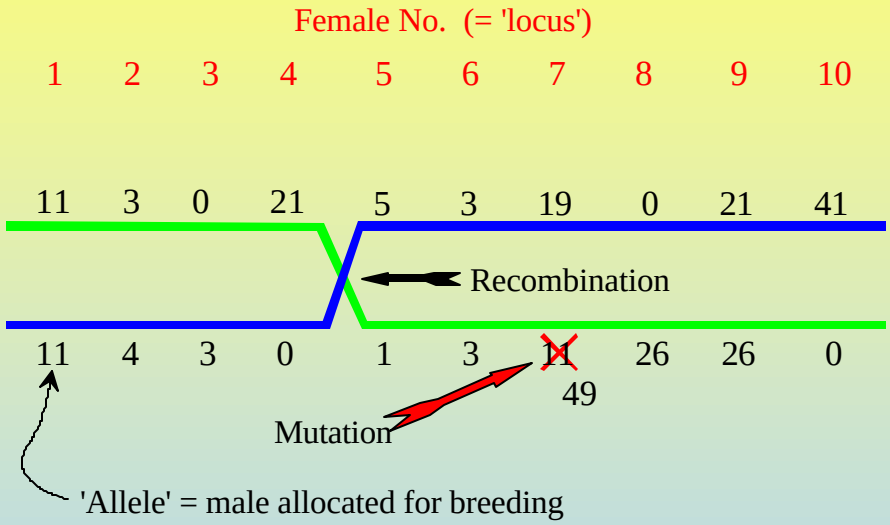
# Genetic Algorithm




- Each individual is a 'genotype' = possible solution
- Each genotype leads to a certain outcome (phenotype)

# Evolutionary algorithms

- Genetic Algorithm
  - 70-ties




- Differential Evolution
  - Price and Storn, 1997: "Dr. Dobbs Journal"
  - Simple and powerful
  - (19 lines of C++ code)

# Differential Evolution

- Propagation is through haploids with only some sex
  - Each candidate is replaced by the best of
    - itself
    - mix of itself and a competitor
- Mutation
  - It perturbates specific variance for each variable!
    i.e. not a ‘standard random noise’ around each variable
- Recombination
  - For each allele change the proportions in the mix
  - Complementary means of creating viable vectors
- Select highest function value

# The DE Algorithm

Sample base population

For generation = 1 to maxgens

Next generation

# The DE Algorithm

Sample base population

For generation = 1 to maxgens

    For individual = 1 to N_individuals

        Draw a challenger (from 3 pop'n members)

        Compare fitness of challenger and individual

        Replace individual if challenger wins

    Next individual

Next generation

# The DE Algorithm

Sample base population

For generation = 1 to maxgens

    For individual = 1 to N_individuals

        Draw a challenger (from 3 pop'n members)

        For j = 1 to N_loci

            Draw allele from challenger or individual

        Next Locus

        Compare fitness of challenger and individual

        Replace individual if challenger wins

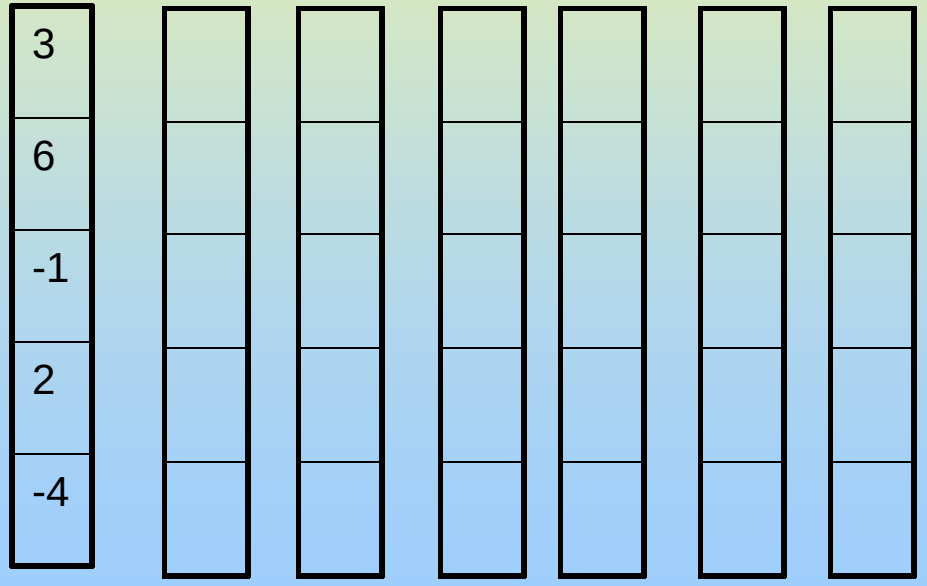
    Next individual

Next generation

# The DE Algorithm

- Sample base population

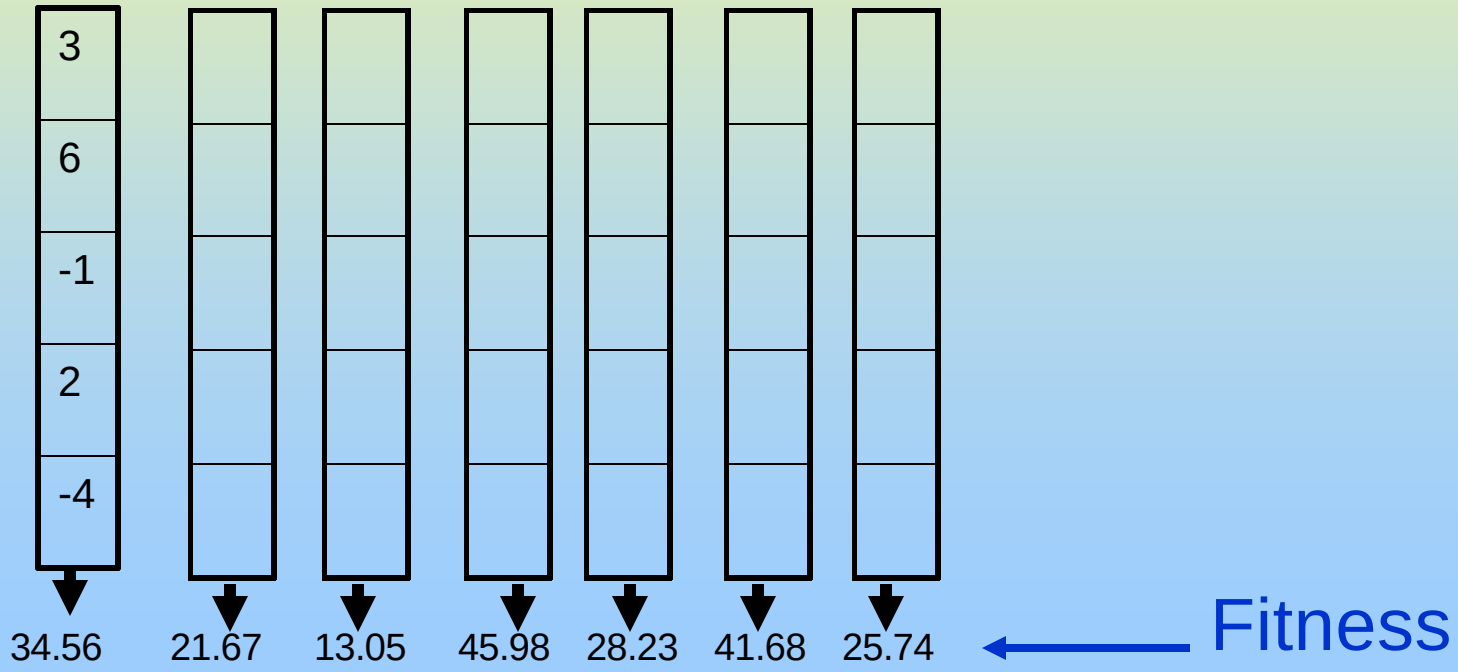
7 individuals




5 loci

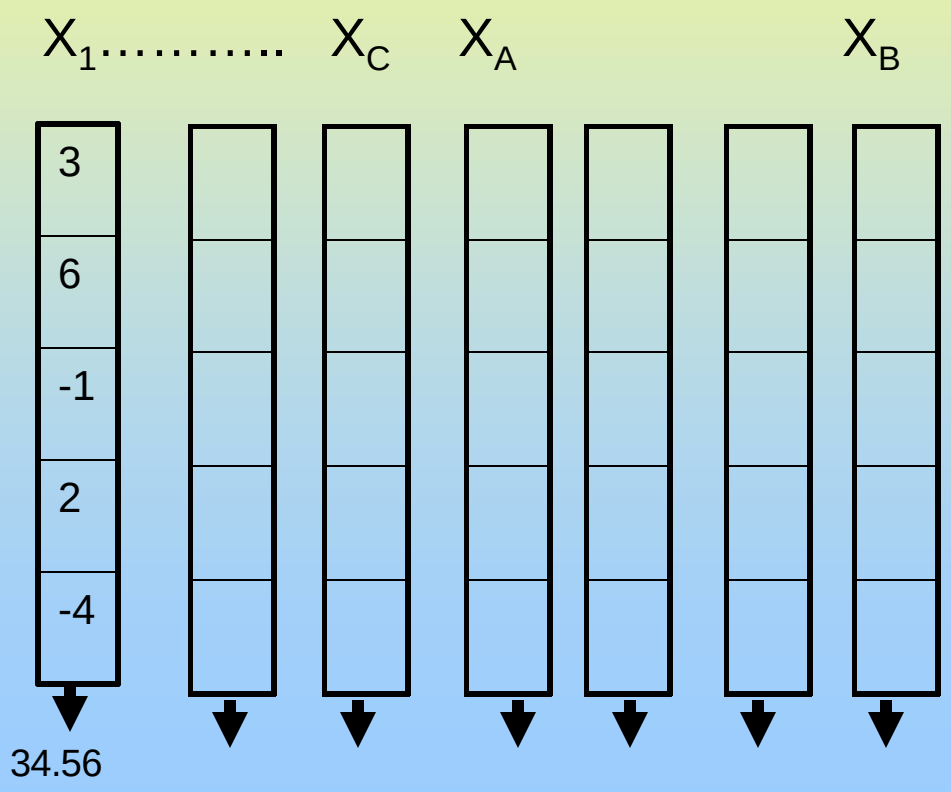
# The DE Algorithm

- Evaluate this generation



# The DE Algorithm

- Next generation: Each individual is challenged




$X_i$ = target

Competes with

$X_T$ = trial,

$X_T = X_C + F.(X_A - X_B)$

Replace $X_i$ with $X_T$

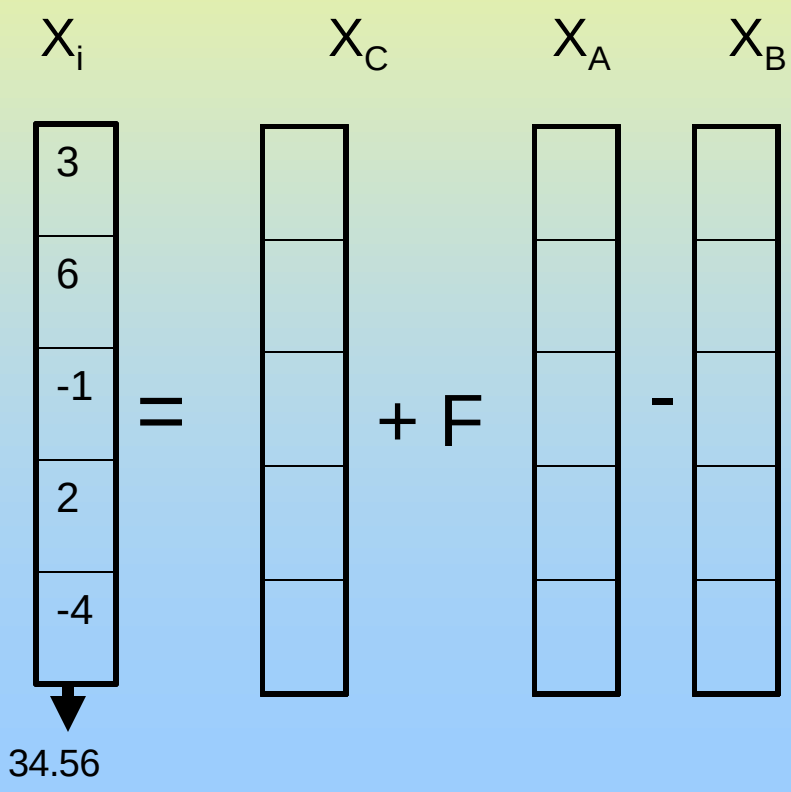
If higher fitness

# The DE Algorithm

$X_i$ = target  Competes with $X_T = X_C + F.(X_A - X_B)$

Mutator

F= 0.4 ~ 1.2




$X_i(i) \rightarrow X_T(i)$

If rnd > CR

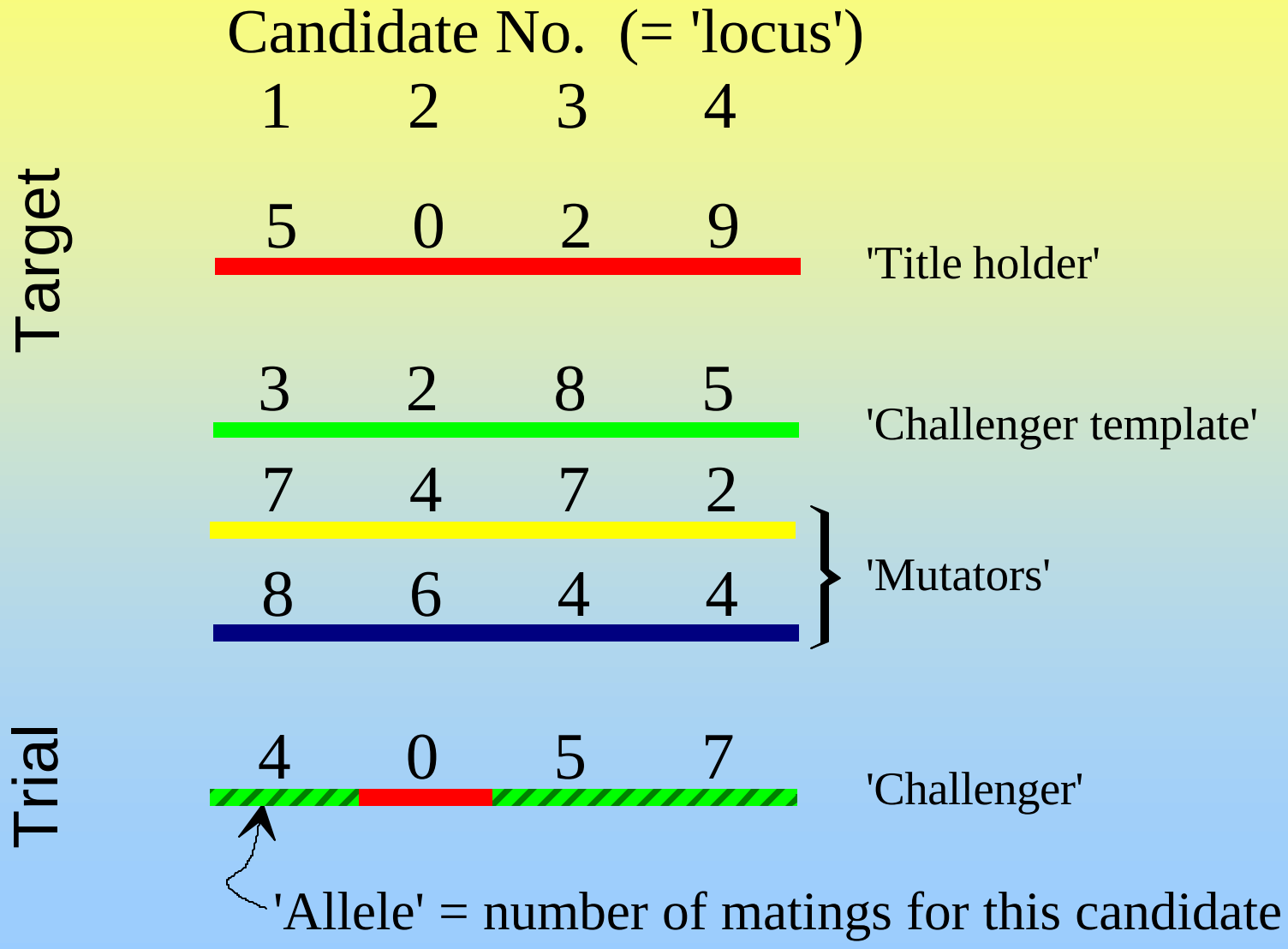
$X_i(i) \rightarrow X_i(i)$

If rnd<CR

Crossover

Sample alleles from target as well as competitor CR ~ 0.8

# Differential Evolution



# Example

Max of $f(x_1,x_2) = 0.2*x_1^2 + 1.45*x_1$
$+0.035*(-0.2*x_2^4 + 3.45*x_2^3 - 5*x_2^2 + 50*x_2 + 3)$

Generation 0

| | x1 | x2 | f(x1,x2) |
|---|---|---|---|
| Indiv 1 | 2.055 | 0.334 | 2.810 |
| Indiv 2 | 0.795 | -2.104 | -4.589 |
| Indiv 3 | 6.220 | 2.747 | 6.978 |
| Indiv 4 | -4.860 | 2.607 | -6.476 |
| Indiv 5 | 3.145 | 2.090 | 6.550 |
| Indiv 6 | -4.546 | 4.112 | 0.011 |
| Indiv 7 | 3.626 | 2.905 | 8.801 |
| Indiv 8 | 2.046 | 4.620 | 15.300 |
| Indiv 9 | 3.714 | -4.438 | -21.747 |
| Indiv 10 | 4.496 | 3.058 | 9.136 |

# Example

Next generation: each individual (target) is challenged (trial), and replaced if trial > target

Generation 0

| x1 | x2 | f(x1,x2) |
|---|---|---|
| 2.055 | 0.334 | 2.810 |
| 0.795 | -2.104 | -4.589 |
| 6.220 | 2.747 | 6.978 |
| -4.860 | 2.607 | -6.476 |
| 3.145 | 2.090 | 6.550 |
| -4.546 | 4.112 | 0.011 |
| 3.626 | 2.905 | 8.801 |
| 2.046 | 4.620 | 15.300 |
| 3.714 | -4.438 | -21.747 |
| 4.496 | 3.058 | 9.136 |

Generation 1

| x1 | x2 | f(x1,x2) |
|---|---|---|
| 2.055 | 0.334 | 2.810 |
| 0.795 | 4.250 | 12.392 |
| 6.220 | 2.747 | 6.978 |
| -4.860 | 2.607 | -6.476 |
| 3.145 | 2.090 | 6.550 |
| -4.546 | 4.112 | 0.011 |
| 3.626 | 2.905 | 8.801 |
| 2.046 | 4.620 | 15.300 |
| 3.714 | 3.775 | 11.919 |
| 4.496 | 3.058 | 9.136 |

# Example

## Generation 8

| x1 | x2 | f(x1,x2) |
|---|---|---|
| 5.356 | 12.125 | 61.575 |
| 4.897 | 12.541 | 61.850 |
| 0.491 | 12.783 | 59.858 |
| 0.491 | 12.123 | 60.208 |
| 7.790 | 12.962 | 57.914 |
| 2.973 | 9.680 | 51.256 |
| -5.128 | 7.638 | 20.550 |
| -0.100 | 12.314 | 59.488 |
| 5.355 | 12.548 | 61.568 |
| 0.490 | 11.565 | 59.156 |

## Generation 16

| x1 | x2 | f(x1,x2) |
|---|---|---|
| 3.359 | 12.495 | 62.195 |
| 3.049 | 12.541 | 62.107 |
| 6.892 | 12.407 | 60.118 |
| 5.354 | 12.491 | 61.614 |
| 5.216 | 12.358 | 61.756 |
| 4.586 | 12.264 | 62.069 |
| 3.534 | 12.237 | 62.242 |
| 3.531 | 12.047 | 62.094 |
| 4.587 | 12.168 | 62.022 |
| 3.818 | 12.347 | 62.256 |

## Generation 24

| x1 | x2 | f(x1,x2) |
|---|---|---|
| 3.636 | 12.331 | 62.264 |
| 3.629 | 12.330 | 62.264 |
| 3.624 | 12.344 | 62.264 |
| 3.619 | 12.322 | 62.264 |
| 3.630 | 12.335 | 62.264 |
| 3.636 | 12.330 | 62.264 |
| 3.633 | 12.336 | 62.264 |
| 3.608 | 12.328 | 62.264 |
| 3.642 | 12.343 | 62.264 |
| 3.607 | 12.336 | 62.264 |

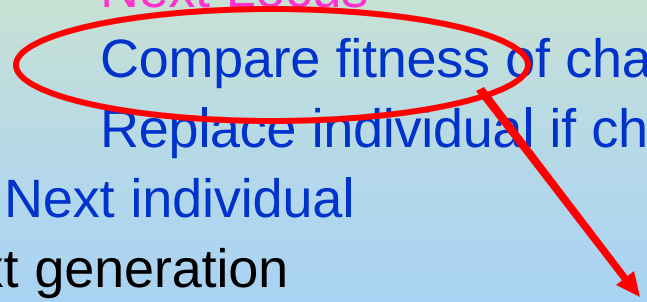
Note: More variance for x1!

# Differential Evolution

- Mutation
  - It perturbates specific variance for each variable! ‘population derived noise’
  - Not a ‘standard random noise’ around each variable
- Recombination
  - ‘knowledge sharing operator’
    - Efficiently shuffle information about successful combination
    - → focus on most promising area of a solution space
  - Complementary means of creating viable vectors
  - Uniform vs non-uniform crossover

# Applying the DE Algorithm

```
Sample base population
For generation = 1 to maxgens
    For individual = 1 to N_individuals
        Draw a challenger (from 3 pop'n members)
        For j = 1 to N_loci
            Draw allele from challenger or individual
        Next Locus
        Compare fitness of challenger and individual
        Replace individual if challenger wins
    Next individual
Next generation
```



The problem is evaluated in a subroutine

# Which Problems are good for a DE to solve?

- Complex
- When $f(x_1, x_2, \ldots x_n)$ is relatively easy to evaluate, but not easy to optimize (need derivatives etc.)

- Problem representation is usually the challenge!
  - Minimize vs maximize
  - Avoid redundancy
  - Efficiently use sample space
  - Can use ranks in allocation problems