

Constraints

- Examples:
 - “A string of numbers should represent a permutation”
(1,2,3) is valid; (1,1,3) is not
 - “The sum of numbers should not be lower than a threshold”
- Possibility 1: fitness function modification
 - setting fitness of unfeasible solutions to zero
(search may be very inefficient due to unfeasible solutions)
 - penalty function (negative terms for violated constraints)
 - barrier function (already penalty if “close to” violation)

Constraints

- Possibility 2 (preferred method): special encoding
 - GA searches always through allowed solutions
 - smaller search space
 - ad hoc method, may be difficult to find
- Example: permutations (see AI course)

Mutations for Permutations

- Insert mutation:
 - Pick two allele values at random
 - Move the second to follow the first, shifting the rest along to accommodate
 - Note: this preserves most of the order and adjacency information; changes the position of numbers a lot

1 2 3 4 5 6 7 8 9



1 2 5 3 4 6 7 8 9

Removed Adjacency: (2,3), (4,5), (5,6)

Added Adjacency: (2,5), (4,6), (5,3)

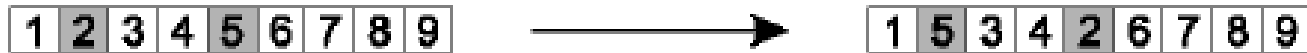
Removed orders: 3->5, 4->5

Added orders: 5->3, 5->4

Changed positions: 3, 4, 5

Mutations for Permutations

- Swap mutation:
 - Pick two alleles at random and swap their positions
 - Disrupts adjacency information and order more; preserves positions



Removed Adjacency:

(1,2), (2,3), (4,5), (5,6)

Added Adjacency:

(1,5), (2,6), (4,2), (5,3)

Removed order:

2->3, 2->4, 2->5, 3->5, 4->5

Added order:

5->3, 5->4, 3->2, 4->2, 5->2

Changed positions:

2, 5

Mutations for Permutations

- Inversion mutation:
 - Pick two alleles at random and then invert the substring between them.
 - Preserves most adjacency information (only breaks two links) but disruptive for order information

1 2 3 4 5 6 7 8 9



1 5 4 3 2 6 7 8 9

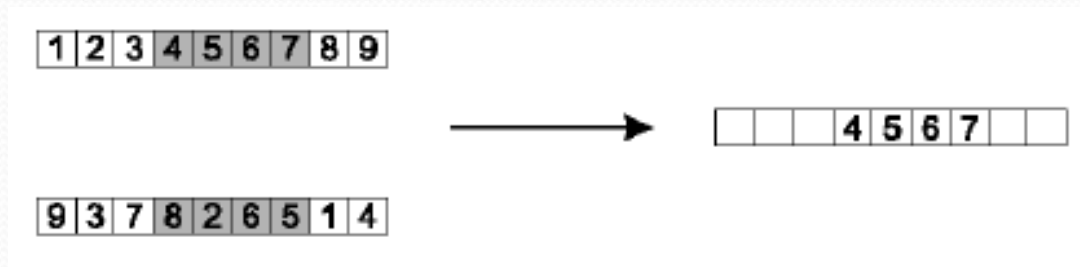
Mutations for Permutations

- Scramble mutation:
 - Pick a subset of genes at random (not necessarily consecutive)
 - Randomly rearrange the alleles in those positions



Crossover for Permutations

- Order one crossover:
 - Choose an arbitrary part from the first parent, copy this part to the first child



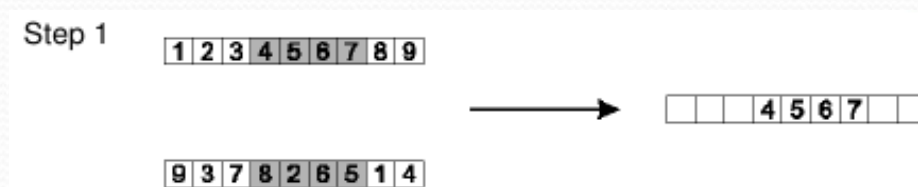
- Copy the numbers that are not in the first part, to the first child:
 - starting right from cut point of the copied part,
 - using the order of the second parent and wrapping around at the end



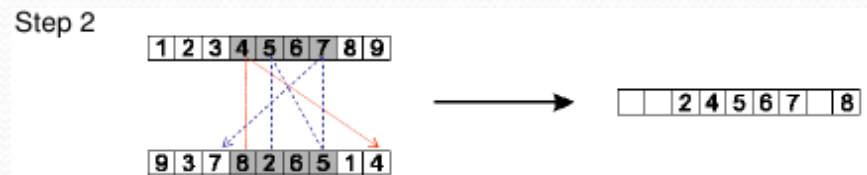
- Analogous for the second child, with parent roles reversed

Crossover for Permutations

- Partially Mapped Crossover (PMX):
 - Choose random segment and copy it from P₁

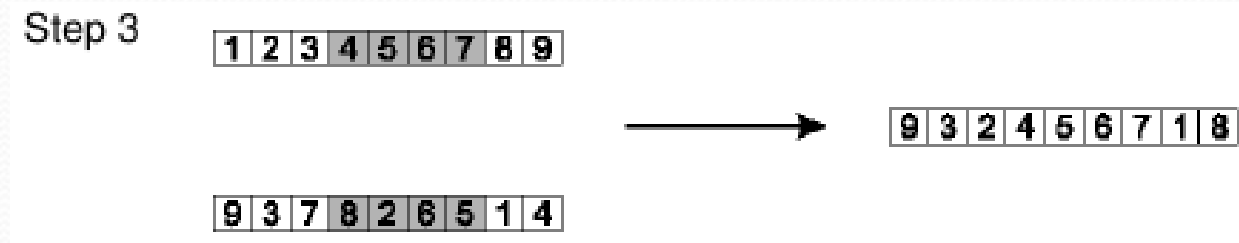


- Starting from the first crossover point look for elements in that segment of P₂ that have not been copied
- For each of these i look in the offspring to see what element j has been copied in its place from P₁
- Place i into the position occupied j in P₂, since we know that we will not be putting j there (as is already in offspring)
- If the place occupied by j in P₂ has already been filled in the offspring k , put i in the position occupied by k in P₂



Crossover for Permutations

- Partially Mapped Crossover (PMX):
 - Having dealt with the elements from the crossover segment, the rest of the offspring can be filled from P₂.



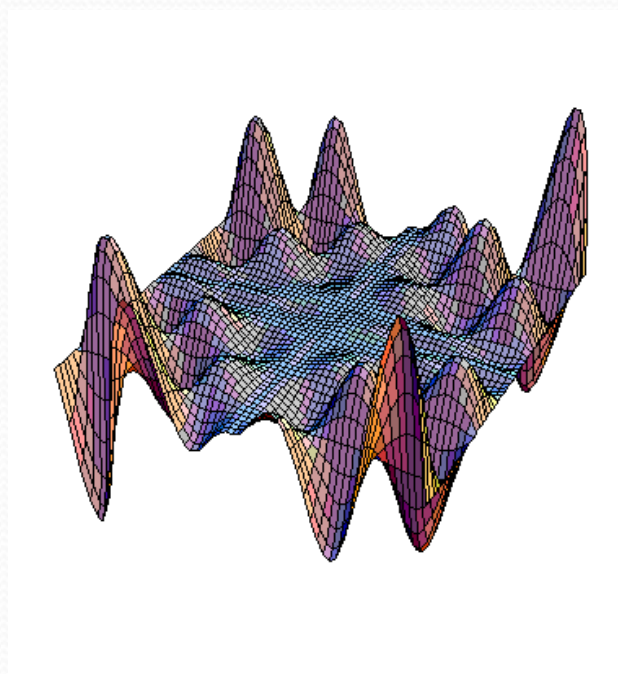
- Idea: maintain position

Order vs Position in Permutations

- Order, but not position of numbers is important in problems such as the traveling salesman problem (visiting all cities in a certain order)
- Position, but not order of numbers is important in problems such as allocating visitors in hotels to rooms (visitors have to be allocated once to one room, but the order of the allocation does not matter)

Evolutionary Strategies

- Numerical optimization problems:
 - **Given** a function f from real numbers to a real number
 - **Find** coordinates at which f is maximized



Evolutionary Strategies

- Main idea:
individuals consist of vectors of real numbers
(not binary)
- Redefinitions of
 - selection
 - crossover
 - mutation
- Operations executed in the order
crossover → mutation → selection

ES: Selection

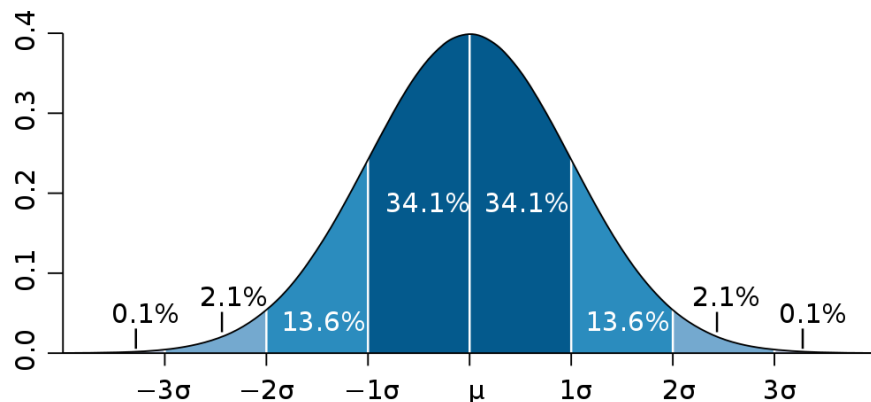
- Not performed *before* mutation and crossover, but after these operations
- It is assumed mutation (& crossover) generate $\lambda > \mu$ individuals (where μ is population size) (typically $\lambda \approx 7\mu$)
- Deterministically eliminate worst individuals from
 - children only: (μ, λ) -ES \rightarrow escapes from local optima more easily (Notational convention)
 - parents and children: $(\mu + \lambda)$ -ES \rightarrow doesn't forget good solutions (“elitist selection”)

ES: Basic Mutation

- An individual is a vector $\vec{h} = (x_1, \dots, x_n)$
- Mutate each x_i by sampling a change from a normal distribution:

$$x_i \leftarrow x_i + \Delta x_i \text{ where } \Delta x_i \sim N(0, \sigma)$$

“sampled from”



Simple modification:
mutation rate for each x_i

Major question:
How to set σ or σ_i ?

MAIN IDEA: make search more efficient
by increasing mutation rate if this seems safe

ES: Basic Mutation

- An algorithm for setting global σ : Improved fitness

- Count the number G_s of successful mutations

- Compute the ratio of successful mutations

$$p_s = G_s / G$$

- Update strategy parameters according to

$$\sigma_i = \begin{cases} \sigma_i / c & \text{if } p_s > 0.2 \\ \sigma_i \cdot c & \text{if } p_s < 0.2 \\ \sigma_i & \text{if } p_s = 0.2 \end{cases}$$

$c \in [0.8, 1.0]$

Increase mutation rate as it appears better solutions are far away
“1/5 rule”

until termination

Basic (1+1) ES

- Common use of the 1/5 rule

```
t := 0;
initialize P(0) := {x̄(0)} ∈ I, I = IRn, x̄ = (x1, ..., xn);
evaluate P(0) : {f(x̄(0))}
while not terminate(P(t)) do
  mutate: x̄'(t) := m(x̄(t))
    where x̄'_i := x_i + σ(t) · N_i(0, 1) ∀i ∈ {1, ..., n}
  evaluate: P'(t) := {x̄'(t)} : {f(x̄'(t))}
  select: P(t + 1) := s(1+1)(P(t) ∪ P'(t));
  t := t + 1;
  if (t mod n = 0) then
    σ(t) := 
$$\begin{cases} \sigma(t - n)/c & , \text{ if } p_s > 1/5 \\ \sigma(t - n) \cdot c & , \text{ if } p_s < 1/5 \\ \sigma(t - n) & , \text{ if } p_s = 1/5 \end{cases}$$

    where ps is the relative frequency of successful
      mutations, measured over intervals of,
      say, 10 · n trials;
    and 0.817 ≤ c ≤ 1;
  else
    σ(t) := σ(t - 1);
  fi
od
```


ES Mutation:

Strategy Parameters

- An individual is a vector $\vec{h} = (x_1, \dots, x_n, \sigma)$
or $\vec{h} = (x_1, \dots, x_n, \sigma_1, \dots, \sigma_n)$
where the σ_i are the standard deviations
- Mutate strategy parameter(s) first
Order is important!
- If the resulting child has high fitness, it is assumed that:
 - quality of phenotype is good
 - quality of strategy parameters that led to this phenotype is good

ES Mutation: Strategy Parameters

- Mutation of one strategy parameter

$$\bar{a} = ((x_1, \dots, x_n), \sigma)$$

$$\bar{a}' = ((x'_1, \dots, x'_n), \sigma')$$

$$\sigma' = \sigma \cdot \exp(\tau_0 \cdot N(0,1))$$

$$x'_i = x_i + \sigma' \cdot N_i(0,1)$$

Individual before mutation

Individual after mutation

1.: Mutation of step sizes

2.: Mutation of objective variables

Here the new σ' is used!

ES Mutation: Strategy Parameters

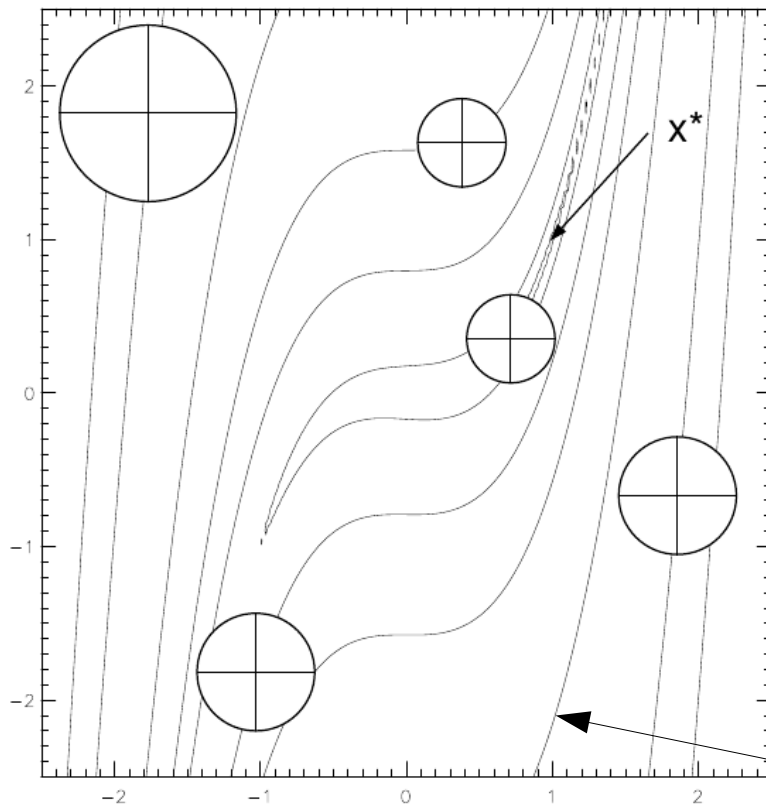
- Here τ_0 is the mutation rate
 - τ_0 bigger: faster but more imprecise
 - τ_0 smaller: slower but more imprecise
- Recommendation for setting τ_0 :

$$\tau_0 = \frac{1}{\sqrt{n}}$$

*H.-P. Schwefel: Evolution and Optimum Seeking, Wiley, NY, 1995.

ES Mutation: Strategy Parameters


⊕ equal probability to place an offspring

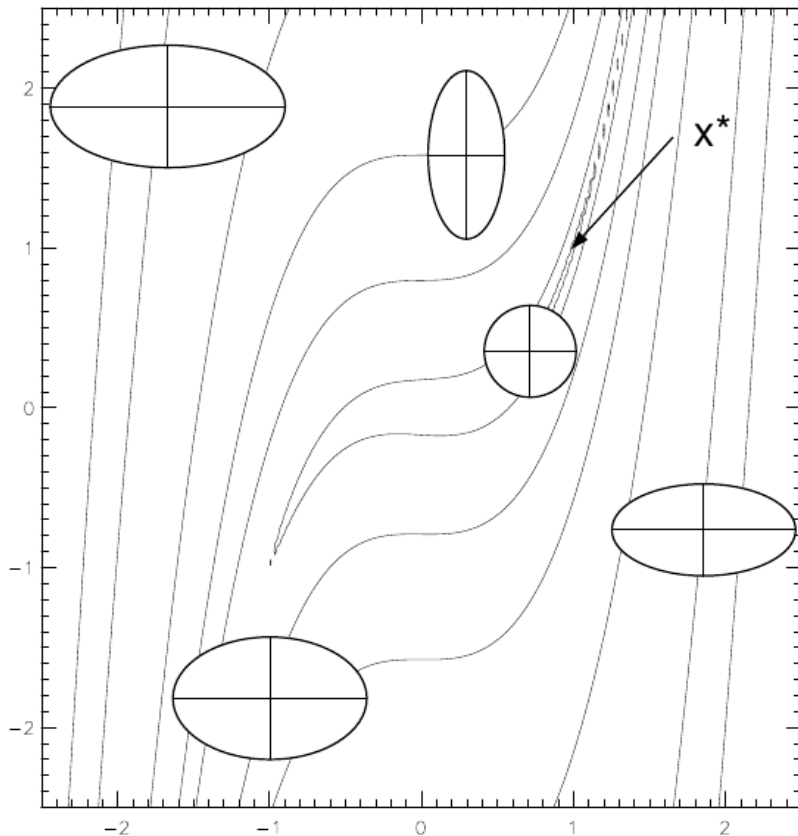


- One parameter for each individual
- 2 dimensional genotype
 $\vec{h} = (x_1, x_2, \sigma)$
- 5 individuals

Line indicates points with equal fitness

ES Mutation: Strategy Parameters

 equal probability to place an offspring



- One parameter for each dimension
- 2 dimensional genotype
 $\vec{h} = (x_1, x_2, \sigma_1, \sigma_2)$
- 5 individuals

ES Mutation: Strategy Parameters

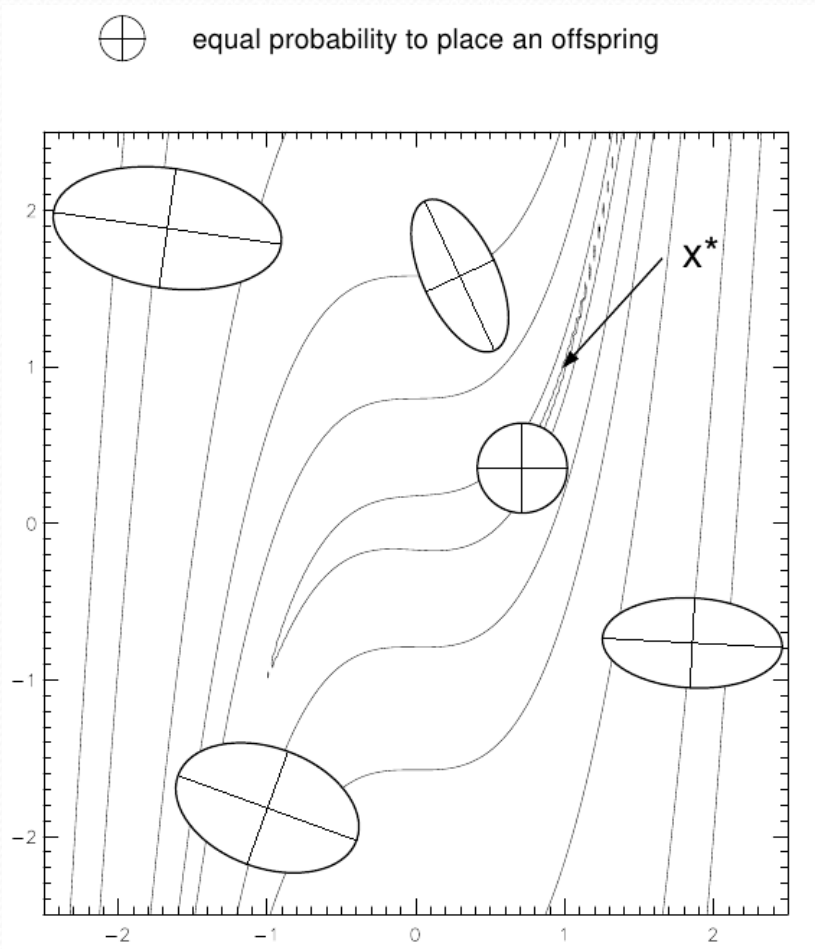
- Mutation of all strategy parameters

$$\begin{aligned}\sigma'_i &= \sigma_i \cdot \exp(\tau' \cdot N(0, 1) + \tau \cdot N_i(0, 1)) \\ x'_i &= x_i + \sigma'_i \cdot N_i(0, 1)\end{aligned}$$

Sample from normal distribution,
the same for all parameters

Update for this specific parameter

ES Mutation: Strategy Parameters



- An individual is a vector
 $\vec{h} = (x_1, \dots, x_n, \sigma_1, \dots, \sigma_n, \alpha_1, \dots, \alpha_m)$

where α_i encode angles

- Also here mutation can be defined
- Mathematical details skipped