How to Build an Evolutionary Algorithm

Brought to you by *P. Adamidis*

The EvoNet Training Committee

Infinite Monkey Theorem

Ένας πίθηκος που χτυπά **τυχαία** πλήκτρα σε μία γραομηχανή για **απεριόριστο** χρονικό διάστημα σχεδόν σίγουρα θα πληκτρολογήσει οποιοδήποτε κείμενο, όπως όλα έργα του William Shakespeare.

Infinite Monkey Theorem (2)

Ωστόσο, η πιθανότητα όλοι οι πίθηκοι που γεμίζουν το παρατηρήσιμο σύμπαν να πληκτρολογήσουν έστω και ένα πλήρες έργο, όπως ο Άμλετ του Σαίξπηρ, είναι τόσο μικρή που η πιθανότητα να συμβεί σε μια χρονική περίοδο εκατοντάδων χιλιάδων τάξεων μεγέθους μεγαλύτερη από την ηλικία του σύμπαντος είναι εξαιρετικά χαμηλή (αλλά όχι μηδενική).

Παράδειγμα: πληκτρολογώντας «banana»

- η γραφομηχανή έχει 50 πλήκτρα
- η πιθανότητα κάθε γράμματος να πληκτρολογείται σωστά είναι 1/50
- η πιθανότητα για σωστή «μπανάνα» είναι (1/50) ^ 6 = μικρότερη από 1 στα 15 δισεκατομμύρια
- Αναμενόμενος αριθμός δοκιμών για να πληκτολογηθεί «μπανάνα» = 15 δισεκατομμύρια

Richard Dawkins Weasel

- Richard Dawkins, “The Blind Watchmaker”, ch 3
- Φράση στόχος: “Methinks it is like a weasel” (28 χαρακτήρες)
- Γραφομηχανή με 27 πλήκτρα και τυχαία πληκτρολόγηση
- Πιθανοί συνδυασμοί: 27^{28} (περίπου 10^{40})
- Εξελικτική αλλαγή: Αρχή με τυχαία επιλογή 28 χαρακτήρων αλλά συνέχιση με επαναλαμβανόμενη αντιγραφή της συμβολοσειράς με κάποιο **λάθος**, κατά την αντιγραφή.
- Εξέταση της νέας φράσης και επιλογή αυτής αν μοιάζει περισσότερο από την αρχική με την φράση στόχο.

Richard Dawkins Weasel

- Χώρος αναζήτησης (Search space): Σύνολο συμβολοσειρών συγκεκριμένου μεγέθους
- Ποιότητα λύσης (Fitness): Πλήθος λαθών στη συμβολοσειρά
- Αλγόριθμος:
  - Δημιουργία αρχικής συμβολοσειράς με τυχαίους χαρακτήρες
  - Επανέλαβε μέχρι να βρεθεί η σωστή συμβολοσειρά:
    - Δημιουργία απογόνου μεταλλάσσοντας ένα χαρακτήρα στη συμβολοσειρά.
    - Εάν η νέα συμβολοσειρά είναι καλύτερη, τότε αντικαθιστά την παλιά συμβολοσειρά

The Steps

In order to build an evolutionary algorithm there are a number of steps that we have to perform:

- Design a representation
- Decide how to initialise a population
- Design a way of mapping a genotype to a phenotype
- Design a way of evaluating an individual

Further Steps

- Design suitable mutation operator(s)
- Design suitable recombination operator(s)
- Decide how to manage our population
- Decide how to select individuals to be parents
- Decide how to select individuals to be replaced
- Decide when to stop the algorithm

Designing a Representation

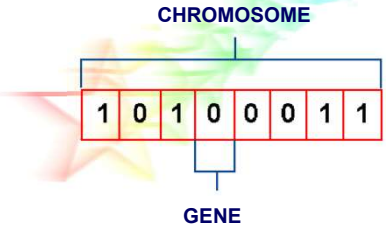
We have to come up with a method of representing an individual as a genotype.

There are many ways to do this and the way we choose must be relevant to the problem that we are solving.

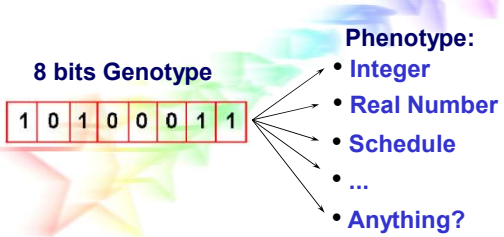
When choosing a representation, we have to bear in mind how the genotypes will be evaluated and what the genetic operators might be

Example: Discrete Representation (Binary alphabet)

- Representation of an individual can be using discrete values (binary, integer, or any other system with a discrete set of values).
- Following is an example of binary representation.

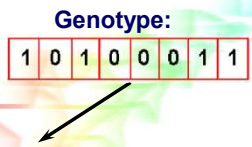


Example: Discrete Representation (Binary alphabet)



Example: Discrete Representation (Binary alphabet)

Phenotype could be integer numbers



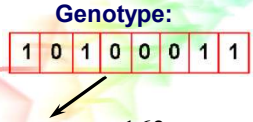
$1*2^7 + 0*2^6 + 1*2^5 + 0*2^4 + 0*2^3 + 0*2^2 + 1*2^1 + 1*2^0 =$

$128 + 32 + 2 + 1 = 163$

Example: Discrete Representation (Binary alphabet)

Phenotype could be Real Numbers

e.g. a number between 2.5 and 20.5 using 8 binary digits



$$x = 2.5 + \frac{163}{256}(20.5 - 2.5) = 13.9609$$

Example: Discrete Representation (Binary alphabet)

Phenotype could be a Schedule
e.g. 8 jobs, 2 time steps

Genotype:

1	0	1	0	0	0	1	1

=

Job	Time Step
1	2
2	1
3	2
4	1
5	1
6	1
7	2
8	2

Phenotype

Example: Real-valued representation

- A very natural encoding if the solution we are looking for is a list of real-valued numbers, then encode it as a list of real-valued numbers! (i.e., not as a string of 1's and 0's)
- Lots of applications, e.g. parameter optimisation

Example: Real valued representation, Representation of individuals

- Individuals are represented as a tuple of n real-valued numbers:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, x_i \in R$$

- The fitness function maps tuples of real numbers to a single real number:

$$f : R^n \rightarrow R$$

Example: Order based representation

- Individuals are represented as permutations
- Used for ordening/sequencing problems
- Famous example: Travelling Salesman Problem where every city gets a assigned a unique number from 1 to *n*. A solution could be (5, 4, 2, 1, 3).
- Needs special operators to make sure the individuals stay valid permutations.

Example: Tree-based representation

- Individuals in the population are trees.
- Any S-expression can be drawn as a tree of functions and terminals.
- These functions and terminals can be anything:
  - Functions: sine, cosine, add, sub, and, If-Then-Else, Turn...
  - Terminals: X, Y, 0.456, true, false, π, Sensor0…
- Example: calculating the area of a circle:

$$\pi * r^2$$

Example: Tree-based representation, Closure & Sufficiency

- We need to specify a function set and a terminal set. It is very desirable that these sets both satisfy closure and sufficiency.
- By closure we mean that each of the functions in the function set is able to accept as its arguments any value and data-type that may possible be returned by some other function or terminal.
- By sufficient we mean that there should be a solution in the space of all possible programs constructed from the specified function and terminal sets.

Initialization

- Uniformly on the search space … if possible
  - Binary strings: 0 or 1 with probability 0.5
  - Real-valued representations: Uniformly on a given interval (OK for bounded values only)
- Seed the population with previous results or those from heuristics. With care:
  - Possible loss of genetic diversity
  - Possible unrecoverable bias

Example: Tree-based representation

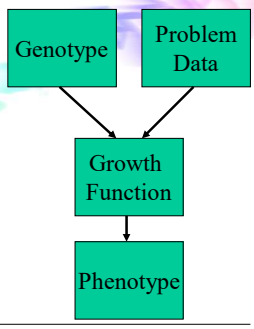
- Pick a function *f* at random from the function set *F*. This becomes the root node of the tree.
- Every function has a fixed number of arguments (unary, binary, ternary, …. , *n*-ary), *z(f)*. For each of these arguments, create a node from either the function set *F* or the terminal set *T*.
- If a terminal is selected then this becomes a leaf
- If a function is selected, then expand this function recursively.
- A maximum depth is used to make sure the process stops.

Example: Tree-based representation, Three Methods

- The Full grow method ensures that every non-back-tracking path in the tree is equal to a certain length by allowing only function nodes to be selected for all depths up to the maximum depth - 1, and selecting only terminal nodes at the lowest level.
- With the Grow method, we create variable length paths by allowing a function or terminal to be placed at any level up to the maximum depth - 1. At the lowest level, we can set all nodes to be terminals.
- Ramp-half-and-half create trees using a variable depth from 2 till the maximum depth. For each depth of tree, half are created using the Full method, and the other half are created using the Grow method.

Getting a Phenotype from our Genotype

- Sometimes producing the phenotype from the genotype is a simple and obvious process.
- Other times the genotype might be a set of parameters to some algorithm, which works on the problem data to produce the phenotype



Evaluating an Individual

- This is by far the most costly step for real applications

  do not re-evaluate unmodified individuals
- It might be a subroutine, a black-box simulator, or any external process

  (e.g. robot experiment)
- You could use approximate fitness - but not for too long

More on Evaluation

- Constraint handling - what if the phenotype breaks some constraint of the problem:
  - penalize the fitness
  - specific evolutionary methods
- Multi-objective evolutionary optimization gives a set of compromise solutions

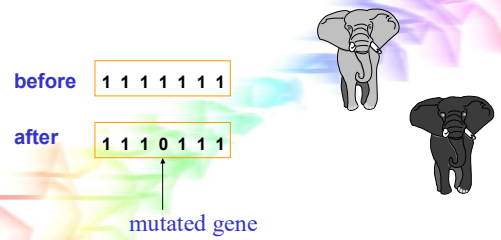
Mutation Operators

We might have one or more mutation operators for our representation.

Some important points are:

- At least one mutation operator should allow every part of the search space to be reached
- The size of mutation is important and should be controllable.
- Mutation should produce valid chromosomes

Example: Mutation for Discrete Representation



Mutation usually happens with probability p_m for each gene

Example: Mutation for real valued representation

Perturb values by adding some random noise

Often, a Gaussian/normal distribution N(0,σ) is used, where

- 0 is the mean value
- σ is the standard deviation

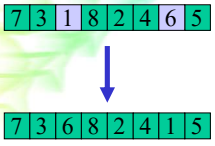
and

$$x'_i = x_i + N(0,\sigma_i)$$

for each parameter

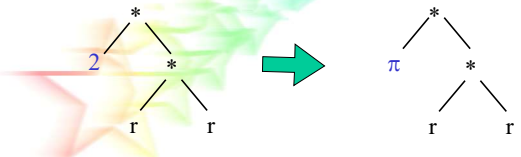
Example: Mutation for order based representation (Swap)

Randomly select two different genes and swap them.



Example: Mutation for tree based representation

Single point mutation selects one node and replaces it with a similar one.



Recombination Operators

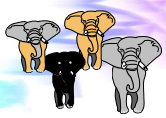
We might have one or more recombination operators for our representation.

Some important points are:

- The child should inherit something from **each** parent. If this is not the case then the operator is a mutation operator.
- The recombination operator should be designed in conjunction with the representation so that recombination is not always catastrophic
- Recombination should produce valid chromosomes

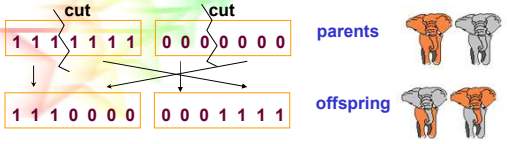
Example: Recombination for Discrete Representation

Whole Population:



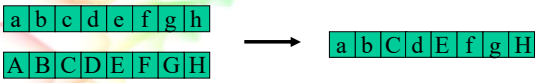
. . .

Each chromosome is cut into n pieces which are recombined. (Example for *n=1*)



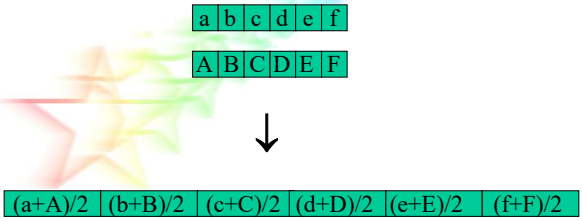
Example: Recombination for real valued representation

Discrete recombination (uniform crossover): given two parents one child is created as follows



Example: Recombination for real valued representation

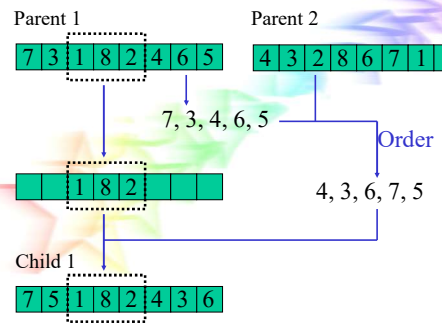
Intermediate recombination (arithmetic crossover): given two parents one child is created as follows



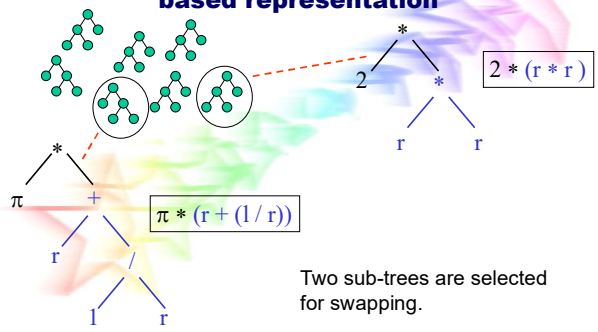
Example: Recombination for order based representation (Order1)

- Choose an arbitrary part from the first parent and copy this to the first child
- Copy the remaining genes that are not in the copied part to the first child:
  - starting right from the cut point of the copied part
  - using the order of genes from the second parent
  - wrapping around at the end of the chromosome
- Repeat this process with the parent roles reversed

Example: Recombination for order based representation (Order1)

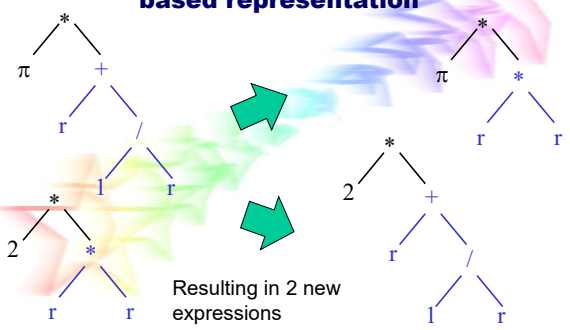


Example: Recombination for tree-based representation



Two sub-trees are selected for swapping.

Example: Recombination for tree-based representation



Selection Strategy

We want to have some way to ensure that better individuals have a better chance of being parents than less good individuals.

This will give us selection pressure which will drive the population forward.

We have to be careful to give less good individuals at least some chance of being parents - they may include some useful genetic material.

Example: Fitness proportionate selection

- Expected number of times f_i is selected for mating is: $f_i / \bar{f}$
- Better (fitter) individuals have:
  - more space
  - more chances to be selected

Example: Fitness proportionate selection

Disadvantages:

- Danger of premature convergence because outstanding individuals take over the entire population very quickly
- Low selection pressure when fitness values are near each other
- Behaves differently on transposed versions of the same function

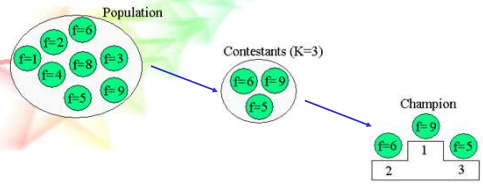
Example: Fitness proportionate selection

Fitness scaling: A cure for FPS

- Start with the raw fitness function f.
- Standardise to ensure:
  - Lower fitness is better fitness.
  - Optimal fitness equals to 0.
- Adjust to ensure:
  - Fitness ranges from 0 to 1.
- Normalise to ensure:
  - The sum of the fitness values equals to 1.

Example: Tournament selection

- Select k random individuals, without replacement
  - k is called the size of the tournament
- Take the best



Example: Ranked based selection

- Individuals are sorted on their fitness value from best to worse. The place in this sorted list is called rank.
- Instead of using the fitness value of an individual, the rank is used by a function to select individuals from this sorted list. The function is biased towards individuals with a high rank (= good fitness).

Example: Ranked based selection

- Fitness: *f(A) = 5, f(B) = 2, f(C) = 19*
- Rank: *r(A) = 2, r(B) = 3, r(C) = 1*

$$h(x) = \min + (\max - \min) * \frac{(r(x)-1)}{n-1}$$

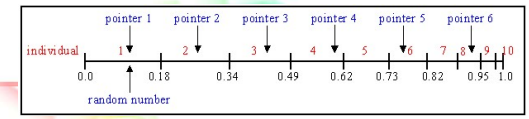
- Function: *h(A) = 3, h(B) = 5, h(C) = 1*
- Proportion on the roulette wheel:
  p(A) = 33.3%, p(B) = 11.1%, p(C) = 55.6%

Example: Stochastic Universal Sampling

- Stochastic universal sampling provides zero bias and minimum spread.
- The individuals are mapped to contiguous segments of a line, such that each individual's segment is equal in size to its fitness exactly as in roulette-wheel selection.
- Equally spaced pointers are placed over the line as many as there are individuals to be selected.
- Consider *NPointer* the number of individuals to be selected, then the distance between the pointers are 1/*NPointer* and the position of the first pointer is given by a randomly generated number in the range [0, 1/*NPointer*].

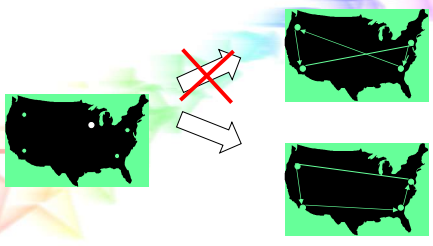
Example: Stochastic Universal Sampling

- For 6 individuals to be selected, the distance between the pointers is 1/6=0.167.
- sample of 1 random number in the range [0, 0.167]: 0.1.



- **After selection the mating population consists of the individuals:**
  - 1, 2, 3, 4, 6, 8.

Traveling Salesperson Problem



Αρχικός πληθυσμός TSP

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

Επιλογή γονέων

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

Επιλογή ευνοώντας τα «καλύτερα» άτομα

Δημιουργία απογόνου
(1 σημείο κοπής)

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

(3,4,5,6,2)

Δημιουργία και άλλου απογόνου

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

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

Μετάλλαξη

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

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

Μετάλλαξη

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

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

Αντικατάσταση
(Απαλοιφή ατόμων)

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

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

Αντικατάσταση των «χειρότερων»

Αντικατάσταση (Εισαγωγή νέων ατόμων)

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

Συνέχεια στην εξέλιξη...

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

Replacement Strategy

The selection pressure is also affected by the way in which we decide which members of the population to kill in order to make way for our new individuals.

We can use the stochastic selection methods in reverse, or there are some deterministic replacement strategies.

We can decide never to replace the best in the population: elitism.

Replacement Strategy (2)

- Generational Replacement
  - Οι απόγονοι αντικαθιστούν συνολικά τους γονείς.
- Steady State Replacement
  - Επιλέγονται μερικά «καλά» (με υψηλή ποιότητα) χρωμοσώματα τα οποία αντικαθιστούν αντίστοιχα «κακά» (με χαμηλή ποιότητα) χρωμοσώματα. Ο υπόλοιπος πληθυσμός επιβιώνει στη νέα γενιά.

Elitism

- Should fitness constantly improves?
  - Re-introduce in the population previous best-so-far (elitism) or
  - Keep best-so-far in a safe place (preservation)
- Theory:
  - GA: preservation mandatory
  - ES: no elitism sometimes is better
- Application: Avoid user's frustration

Recombination vs Mutation

- Recombination
  - modifications depend on the whole population
  - decreasing effects with convergence
  - exploitation operator
- Mutation
  - mandatory to escape local optima
  - strong causality principle
  - exploration operator

Recombination vs Mutation (2)

- Historical “irrationale”
  - GA emphasize crossover
  - ES and EP emphasize mutation
- Problem-dependent rationale:
  - fitness partially separable?
  - existence of building blocks?
  - Semantically meaningful recombination operator?

Use recombination if useful!

Stopping criterion

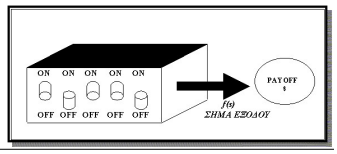
- The optimum is reached!
- Limit on CPU resources:

  Maximum number of fitness evaluations
- Limit on the user’s patience:

  After some generations without improvement

Ένας Απλός Γενετικός Αλγόριθμος (1)

- Το Πρόβλημα Βελτιστοποίησης του Μαύρου Κουτιού με τους 5 Διακόπτες.
- Για κάθε συνδυασμό θέσεων των διακοπτών υπάρχει ένα σήμα εξόδου *f(s)*, όπου s είναι ένας συγκεκριμένος συνδυασμός των θέσεων των πέντε διακοπτών.
- Το ζητούμενο του προβλήματος είναι να θέσουμε τους διακόπτες έτσι ώστε να αποκομίσουμε τη μέγιστη δυνατή τιμή της *f*.



Ένας Απλός Γενετικός Αλγόριθμος (2)

- Οι ΓΑ (όπως και γενικότερα οι ΕΑ) δεν χρειάζεται να γνωρίζουν τον τρόπο λειτουργίας του μαύρου κουτιού.
- Εκείνο που χρειάζονται είναι:
  - μία μέθοδος κωδικοποίησης και
  - ένας τρόπος αξιολόγησης των κωδικοποιήσεων.
- Με άλλες μεθόδους βελτιστοποίησης είναι πιθανό να δουλεύαμε απευθείας με το σύνολο των παραμέτρων (δηλαδή, τους συνδυασμούς των διακοπτών).
- Με τους ΓΑ, το πρώτο που κάνουμε είναι να κωδικοποιήσουμε τους διακόπτες ως μία συμβολοσειρά πεπερασμένου μήκους.
- Μια σειρά από μονάδες και μηδενικά μήκους πέντε, όπου καθένας από τους πέντε διακόπτες αναπαριστάται από μία μονάδα αν είναι ανοιχτός, και από ένα μηδέν αν είναι κλειστός (πχ. 10110)

Ένας Απλός Γενετικός Αλγόριθμος (3)

- Ένας ΓΑ ξεκινά με έναν πληθυσμό συμβολοσειρών (άτομα).
- Ακολούθως δημιουργεί συνεχώς νέους βελτιωμένους πληθυσμούς συμβολοσειρών.
- Στο συγκεκριμένο πρόβλημα των πέντε διακοπτών, μία τυχαία εκκίνηση με το στρίψιμο ενός νομίσματος (1 = κεφαλή, 0 = γράμματα) μπορεί να δημιουργήσει τον αρχικό πληθυσμό μεγέθους n = 4 (που είναι πολύ μικρός για τα πρότυπα των ΓΑ).

  01101
  11000
  01000
  10011
- Εκτός της αρχικοποίησης του πληθυσμού, πρέπει να ορίσουμε ένα σύνολο από απλές λειτουργίες, οι οποίες παίρνουν αυτόν τον αρχικό πληθυσμό και “γεννούν” επιτυχημένους πληθυσμούς οι οποίοι βελτιώνονται με το χρόνο.

Ένας Απλός Γενετικός Αλγόριθμος (4)

- Ένας απλός ΓΑ, στηρίζεται σε τρεις τελεστές:
  - *Επιλογή*,
  - *Διασταύρωση (Ανασυνδυασμός)*, και
  - *Μετάλλαξη*.
- Κατά την *αναπαραγωγή*, ξεχωριστές συμβολοσειρές (άτομα) επιλέγονται σύμφωνα με τις τιμές ποιότητας που τους έχει ανατεθεί από την συνάρτησης ποιότητας *f* (οι βιολόγοι καλούν αυτή τη συνάρτηση, *συνάρτηση προσαρμογής-καταλληλότητας*).
- Μπορούμε να σκεφτόμαστε τη συνάρτηση ποιότητας *f* ως κάποιο μέσο μέτρησης του κέρδους, της χρησιμότητας της ποιότητας ή της καταλληλότητας που επιθυμούμε να βελτιστοποιήσουμε.
- Η επιλογή συμβολοσειρών ανάλογα με τις τιμές ποιότητάς τους σημαίνει ότι, σειρές με μια υψηλότερη τιμή έχουν και υψηλότερη πιθανότητα συνεισφοράς ενός ή περισσοτέρων απογόνων στην επόμενη γενιά. Αυτός ο χειρισμός είναι μία τεχνητή έκδοση της φυσικής επιλογής,

Ένας Απλός Γενετικός Αλγόριθμος (5)

- Ο τελεστής της *επιλογής* μπορεί να υλοποιηθεί σε αλγοριθμική μορφή με πολλούς τρόπους. Ένας από αυτούς είναι να δημιουργήσουμε έναν "μεροληπτικό" τροχό ρουλέτας
- Ας υποθέσουμε ότι το δείγμα του πληθυσμού των τεσσάρων ατόμων στο πρόβλημα του μαύρου κουτιού έχει τις τιμές ποιότητας, που βλέπουμε στον Πίνακα. (Προς το παρόν τις δεχόμαστε σαν τιμές -τη συνάρτηση και την κωδικοποίηση που τις δημιουργούν θα τις εξετάσουμε παρακάτω.)

No..	*Σειρά*	*Ποιότητα*	*% του Συνόλου*
1	01101	169	14.4
2	11000	576	49.2
3	01000	64	5.5
4	10011	361	30.9
Σύνολο:		1170	100.0

Ένας Απλός Γενετικός Αλγόριθμος (6)

- Ο τροχός της ρουλέτας που προκύπτει για την αναπαραγωγή αυτής της γενεάς μοιράζεται σε τέσσερα μέρη με αντιστοιχία ανάλογη, για κάθε άτομο, των ποσοστών που βλέπουμε στη στήλη "% του Συνόλου".
- Για να επιλέξουμε τους γονείς οι οποίοι θα αναπαραχθούν, απλά γυρίζουμε τον τροχό της ρουλέτας, όπως χωρίστηκε, τέσσερις φορές.
- Κάθε φορά που χρειαζόμαστε ακόμη έναν απόγονο, ένα απλό γύρισμα του σταθμισμένου τροχού αποφέρει ένα άτομο το οποίο θα λειτουργήσει ως γονέας για την αναπαραγωγή κάποιου παιδιού.
- Με αυτό τον τρόπο, τα άτομα με καλύτερη ποιότητα έχουν πιο πολλούς απόγονους στην νέα γενεά.

No..	*Σειρά*	*Ποιότητα*	*% του Συνόλου*
1	01101	169	14.4
2	11000	576	49.2
3	01000	64	5.5
4	10011	361	30.9
Σύνολο:		1170	100.0

Ένας Απλός Γενετικός Αλγόριθμος (7)

- Μετά την επιλογή, μία απλή διασταύρωση μπορεί να προχωρήσει σε δύο βήματα:
  - Μέλη των ατόμων που έχουν επιλεγεί, συνταιριάζονται στην τύχη.
  - Κάθε ζευγάρι ατόμων υφίσταται μία διασταύρωση ως εξής: επιλέγεται ομοιόμορφα μία ακέραιη θέση *k* κατά μήκος της σειράς, μέσα στο πεδίο [1, *l* - 1] (όπου *l* το μήκος της σειράς) με τυχαίο τρόπο. Δύο νέες σειρές δημιουργούνται ανταλλάσσοντας όλους τους χαρακτήρες μεταξύ των θέσεων *k* + 1 και *l* συνολικά.
- Για παράδειγμα, σκεφθείτε τις σειρές A1 και A2 από τον αρχικό πληθυσμό του παραδείγματός μας.

  A1 = 0110|1
  A2 = 1100|0

  - Ας υποθέσουμε ότι επιλέγουμε έναν τυχαίο αριθμό, *k* = 4.
- Η διασταύρωση στην οποία καταλήγουμε, αποφέρει δύο νέες σειρές:

  A′1 = 01100
  A′2 = 11001

Ένας Απλός Γενετικός Αλγόριθμος (8)

- Η μηχανική της αναπαραγωγής και της διασταύρωσης είναι εκπληκτικά απλή, συμπεριλαμβάνοντας τυχαίους αριθμούς γενεών, αντίγραφα ατόμων/συμβολοσειρών, και κάποιες ανταλλαγές τμημάτων ατόμων.
- Παρ' όλα αυτά, η συνδυασμένη έμφαση της αναπαραγωγής και η δομημένη, αν και τυχαία δημιουργημένη, ανταλλαγή πληροφοριών της διασταύρωσης, δίνουν στους ΓΑ ένα μεγάλο μέρος από τη δύναμή τους.
- Η επιλογή και η διασταύρωση συνδυάζουν στην αναζήτηση, την παροχή ενδεχόμενων νέων ιδεών. Αυτή η εμπειρία έμφασης και διασταύρωσης είναι ανάλογη της ανθρώπινης αλληλεπίδρασης (έννοιες, αντιλήψεις) των παρελθόντων δοκιμών.

Ένας Απλός Γενετικός Αλγόριθμος (9)

- Αν, όμως, ο συνδυασμός αναπαραγωγής και διασταύρωσης δίνει στους ΓΑ το κύριο μέρος της διαδικαστικής τους δύναμης, τότε ποιος είναι ο σκοπός του τελεστή μετάλλαξης;
- Η μετάλλαξη διαδραματίζει ένα δευτερεύοντα ρόλο στη λειτουργία των ΓΑ. Η μετάλλαξη είναι αναγκαία διότι, παρ' όλο που η αναπαραγωγή και η διασταύρωση αναζητούν αποτελεσματικά και επανασυνδυάζουν τις υπάρχουσες έννοιες, μπορεί να χάσουν κάποιο, ενδεχόμενα, χρήσιμο γενετικό υλικό (μονάδες ή μηδενικά σε συγκεκριμένες θέσεις).
- Στον απλό ΓΑ η μετάλλαξη εκφράζεται, ως μία περιστασιακή (και με μικρή πιθανότητα) τυχαία μετατροπή της τιμής μιας θέσης, σε κάποια συμβολοσειρά/άτομο. Στη δυαδική κωδικοποίηση του προβλήματος του μαύρου κουτιού, αυτό μεταφράζεται σε αλλαγή μίας μονάδας σε μηδέν, και αντίστροφα.

Ένας Απλός Γενετικός Αλγόριθμος (10)

- Ο τελεστής μετάλλαξης παίζει ένα δευτερεύοντα ρόλο στους ΓΑ.
  - η συχνότητα της μετάλλαξης για να αποκτήσουμε καλά αποτελέσματα στις εμπειρικές μελέτες ΓΑ είναι του επιπέδου μίας μετάλλαξης ανά χίλια bit (θέσεις).
- Οι τιμές μετάλλαξης είναι συνήθως μικρότερες σε φυσικούς πληθυσμούς, οδηγώντας μας στο συμπέρασμα ότι η μετάλλαξη αντιμετωπίζεται σαν ένας δευτερεύον μηχανισμός της προσαρμογής του ΓΑ.
- Οι τρεις μηχανισμοί που εξετάστηκαν (επιλογή, διασταύρωση, μετάλλαξη) είναι εξίσου απλοί υπολογιστικά και αποτελεσματικοί στην αντιμετώπιση ενός μεγάλου αριθμού από σημαντικά προβλήματα βελτιστοποίησης.

Προσομοίωση Γενετικού Αλγόριθμου (1)

- Έστω το πρόβλημα μεγιστοποίησης της συνάρτησης
  $f(x) = x^2$,
  όπου το x επιτρέπεται να παίρνει τιμές μεταξύ 0 και 31.
- Για να χρησιμοποιήσουμε το ΓΑ πρέπει πρώτα να κωδικοποιήσουμε τις παραμέτρους του προβλήματός μας ως μία βέλτιστου μήκους δυαδική συμβολοσειρά.
- Στο δεδομένο πρόβλημα θα κωδικοποιήσουμε τη μεταβλητή x ως ένα δυαδικό μη-προσημασμένο ακέραιο, μήκους 5. Γιατί 5;
- Με μία καλά ορισμένη συνάρτηση ποιότητας και την κατάλληλη κωδικοποίηση, προσομοιώνουμε μία μοναδική γενεά ενός ΓΑ με επιλογή, διασταύρωση και μετάλλαξη.

Προσομοίωση Γενετικού Αλγόριθμου (2)

- Επιλέγουμε έναν αρχικό πληθυσμό στην τύχη.
- Μέγεθος πληθυσμού 4 (ρίχνοντας ένα νόμισμα 20 φορές όπως στο παράδειγμα του μαύρου κουτιού).

Νούμερο Σειράς / Ατόμου	Αρχικός Πληθυσμός (Τυχαία Δημιουργημένος)	Τιμή του x (Μη Προσημασμένος Ακέραιος)	$f(x) = x^2$	$pselect_i$ $f_i / \Sigma f$	Αναμενόμενη Μέτρηση $f_i / \bar{f}$	Πραγματική Μέτρηση (από τον τροχό της ρουλέτας)
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Άθροισμα:			1170	1.00	4.00	4.0
Μέσος:			293	0.25	1.00	1.0
Μέγιστο:			576	0.49	1.97	2.0

Προσομοίωση Γενετικού Αλγόριθμου (3)

- Με βάση τον παραπάνω κοινό χώρο από σειρές που ψάχνουν για τα ταίρια τους, η απλή διασταύρωση προχωράει σε δύο βήματα:
  - οι σειρές/άτομα συνταιριάζονται τυχαία,
  - τα ζευγάρια αυτά των σειρών/ατόμων διασταυρώνονται με τυχαία επιλογή των θέσεων διασταύρωσης.

Κοινός Συνδυαστικός Χώρος	Ταίρι (Τυχαία Επιλεγμένο)	Σημείο Διασταύρωσης (Τυχαία Επιλεγμένο)	Νέος Πληθυσμός
0 1 1 0 | 1	2	4	0 1 1 0 0
1 1 0 0 | 0	1	4	1 1 0 0 1
1 1 | 0 0 0	4	2	1 1 0 1 1
1 0 | 0 0 1	3	2	1 0 0 0 0

Προσομοίωση Γενετικού Αλγόριθμου (4)

- Ο τελευταίος τελεστής, η μετάλλαξη, αποδίδεται σε μία bit προς bit βάση.
- Υποθέτουμε ότι η πιθανότητα μετάλλαξης σε αυτόν τον έλεγχο είναι 0.001.
- Με 20 θέσεις ψηφίων θα έπρεπε να περιμένουμε $20 \times 0.001 = 0.02$ bits να υποστούν μετάλλαξη κατά τη διάρκεια μιας δοσμένης γενεάς.
- Είναι λοιπόν εύκολα αντιληπτό ότι για την προσομοίωση αυτής της διαδικασίας, δεν υποβάλλονται κάποια bits σε μετάλλαξη.
- Άρα σε αυτή τη γενεά δεν έχουμε καμία αλλαγή από 0 σε 1 σε κάποια θέση ψηφίου από την εφαρμογή του τελεστή μετάλλαξης.

Προσομοίωση Γενετικού Αλγόριθμου (3)

- Αφού έλαβαν χώρα οι λειτουργίες και των τριών τελεστών του ΓΑ, ο νέος πληθυσμός είναι έτοιμος για έλεγχο.
- Για να γίνει αυτό:
  - αποκωδικοποιούμε τις νέες σειρές που δημιουργήθηκαν από το ΓΑ
  - υπολογίζουμε τις τιμές της συνάρτησης ποιότητας από τις τιμές x που αποκωδικοποιήθηκαν.

Κοινός Συνδυαστικός Χώρος	Ταίρι (Τυχαία Επιλεγμένο)	Σημείο Διασταύρωσης (Τυχαία Επιλεγμένο)	Νέος Πληθυσμός	Τιμή x	$f(x) = x^2$
0 1 1 0 | 1	2	4	0 1 1 0 0	12	144
1 1 0 0 | 0	1	4	1 1 0 0 1	25	625
1 1 | 0 0 0	4	2	1 1 0 1 1	27	729
1 0 | 0 0 1	3	2	1 0 0 0 0	16	256
					1754
					439
					729

Προσομοίωση Γενετικού Αλγόριθμου (5)

- Αφού έλαβαν χώρα οι λειτουργίες και των τριών τελεστών του ΓΑ, ο νέος πληθυσμός είναι έτοιμος για έλεγχο.
- Για να γίνει αυτό, αποκωδικοποιούμε τις νέες σειρές που δημιουργήθηκαν από το ΓΑ και υπολογίζουμε τις τιμές της συνάρτησης ποιότητας από τις τιμές x που αποκωδικοποιήθηκαν.

Algorithm performance

- Never draw any conclusion from a single run
  - use statistical measures (averages, medians)
  - from a sufficient number of independent runs
- From the application point of view
  - design perspective:
    find a very good solution at least once
  - production perspective:
    find a good solution at almost every run

Algorithm Performance (2)

Remember the WYTIWYG principal:

"What you test is what you get" - don´t tune algorithm performance on toy data and expect it to work with real data.

Key issues

Genetic diversity

- differences of genetic characteristics in the population
- loss of genetic diversity = all individuals in the population look alike
- snowball effect
- convergence to the nearest local optimum
- in practice, it is irreversible

Key issues (2)

Exploration vs Exploitation

- Exploration =sample unknown regions
- Too much exporation = random search, no convergence
- Exploitation = try to improve the best-so-far individuals
- Too much expoitation = local search only … convergence to a local optimum

Παράλληλοι/Κατανεμημένοι Εξελικτικοί Αλγόριθμοι

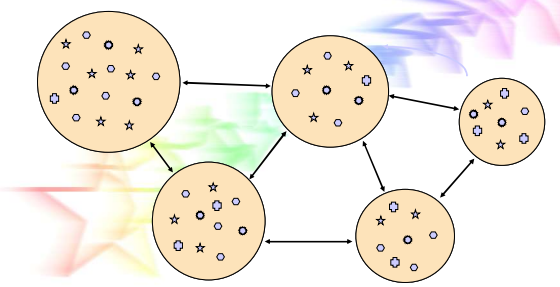
Δύο λόγοι (κυρίως):

- Βελτίωση υπολογιστικής απόδοσης (πόροι και ταχύτητα)
  - προσθέτοντας επεξεργαστές, μνήμη και δίκτυα διασύνδεσης και βάζοντάς τα να συνεργαστούν σε ένα δεδομένο πρόβλημα.
- Η τροποποίηση της λειτουργίας του αλγορίθμου μπορεί επίσης να βοηθήσει στην εξεύρεση λύσεων.

Μοντέλα Παραλληλισμού

- Η κατάταξη των αρχιτεκτονικών υπολογιστικών συστημάτων του Flynn είναι ακόμη ευρέως αποδεκτή.
- Ταξινόμηση ανάλογα με τις ακολουθίες εντολών (instruction streams) που εκτελούνται ταυτόχρονα σε ακολουθίες δεδομένων (data streams)
  - SISD: Single instruction, Single Data stream.
  - SIMD: Single Instruction, Multiple Data stream (the preferred model).
  - MISD: Multiple instruction, single data stream.
  - MIMD: Multiple instruction, multiple data stream.

Island Model



Island Model - Pseudocode

1: Initialize a population made up of subpopulations or islands, $P^{(0)} = \{P_1^{(0)}, \ldots, P_m^{(0)}\}$.
2: Let $t := 1$.
3: **loop**
4: **for** each island i **do in parallel**
5: **if** $t \bmod \tau = 0$ **then**
6: Send selected individuals from island $P_i^{(t)}$ to selected neighboring islands.
7: Receive immigrants $I_i^{(t)}$ from islands for which island $P_i^{(t)}$ is a neighbor.
8: Replace $P_i^{(t)}$ by a subpopulation resulting from a selection among $P_i^{(t)}$ and $I_i^{(t)}$.
9: **end if**
10: Produce $P_i^{(t+1)}$ by applying reproduction operators and selection to $P_i^{(t)}$.
11: **end for**
12: Let $t := t + 1$.
13: **end loop**

Important terms

- Emigration policy (leaving)
- Immigration policy (incoming)
- Migration interval or Migration frequency
- Number of migrants
- Migration topology