Genetic Algorithms

What You Will Learn From This Tutorial?

- What is a genetic algorithm?
- Principles of genetic algorithms.
- How to design an algorithm?
- Comparison of GA and conventional algorithms.

- Mathematics behind GA-s

- Applications of GA
  - GA and the Internet
  - Genetic search based on multiple mutation approaches

Part I: GA Theory

What are genetic algorithms?

How to design a genetic algorithm?

Genetic Algorithm Is Not...

...Gene coding

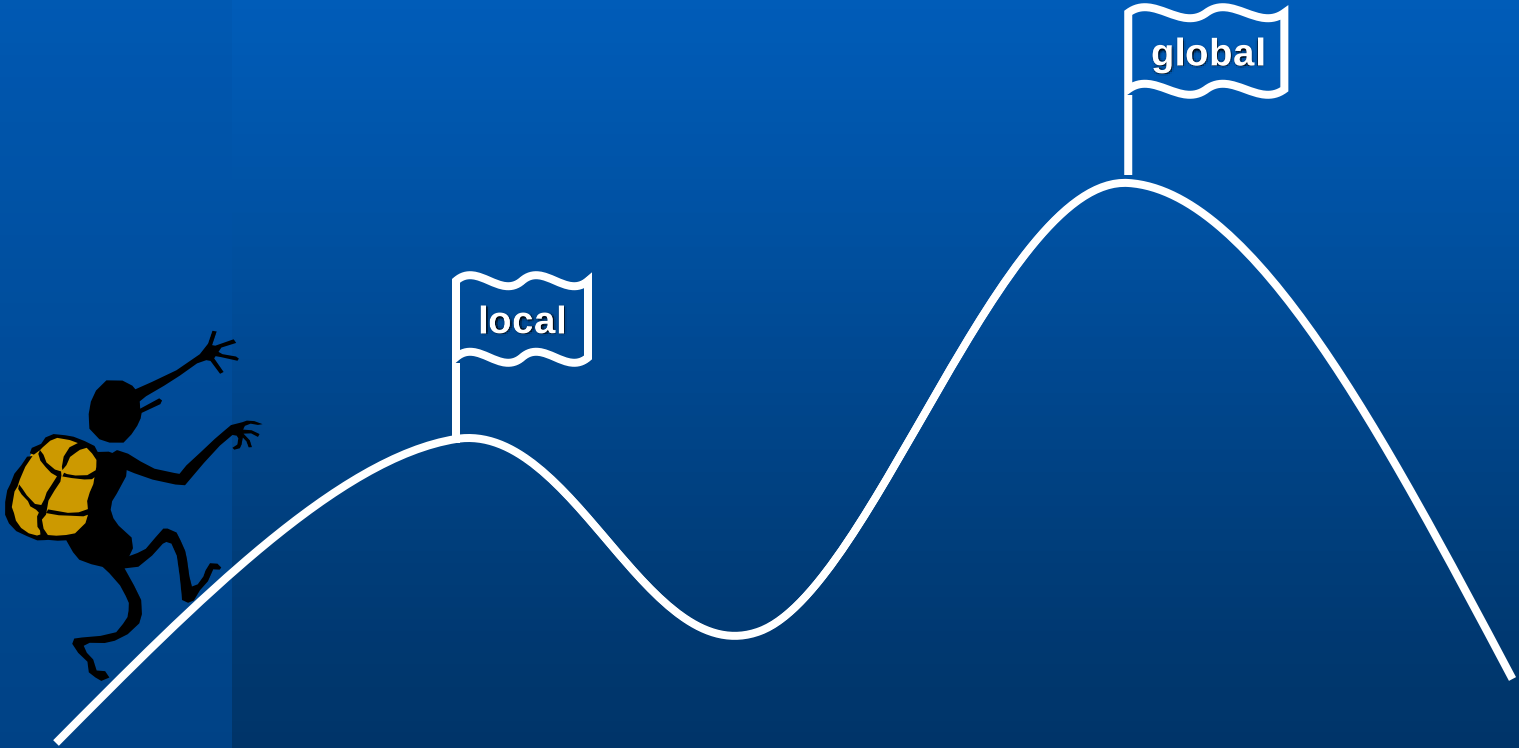
Genetic Algorithm Is...

… Computer algorithm

That resides on principles of genetics and evolution

Instead of Introduction...

- **Hill climbing**

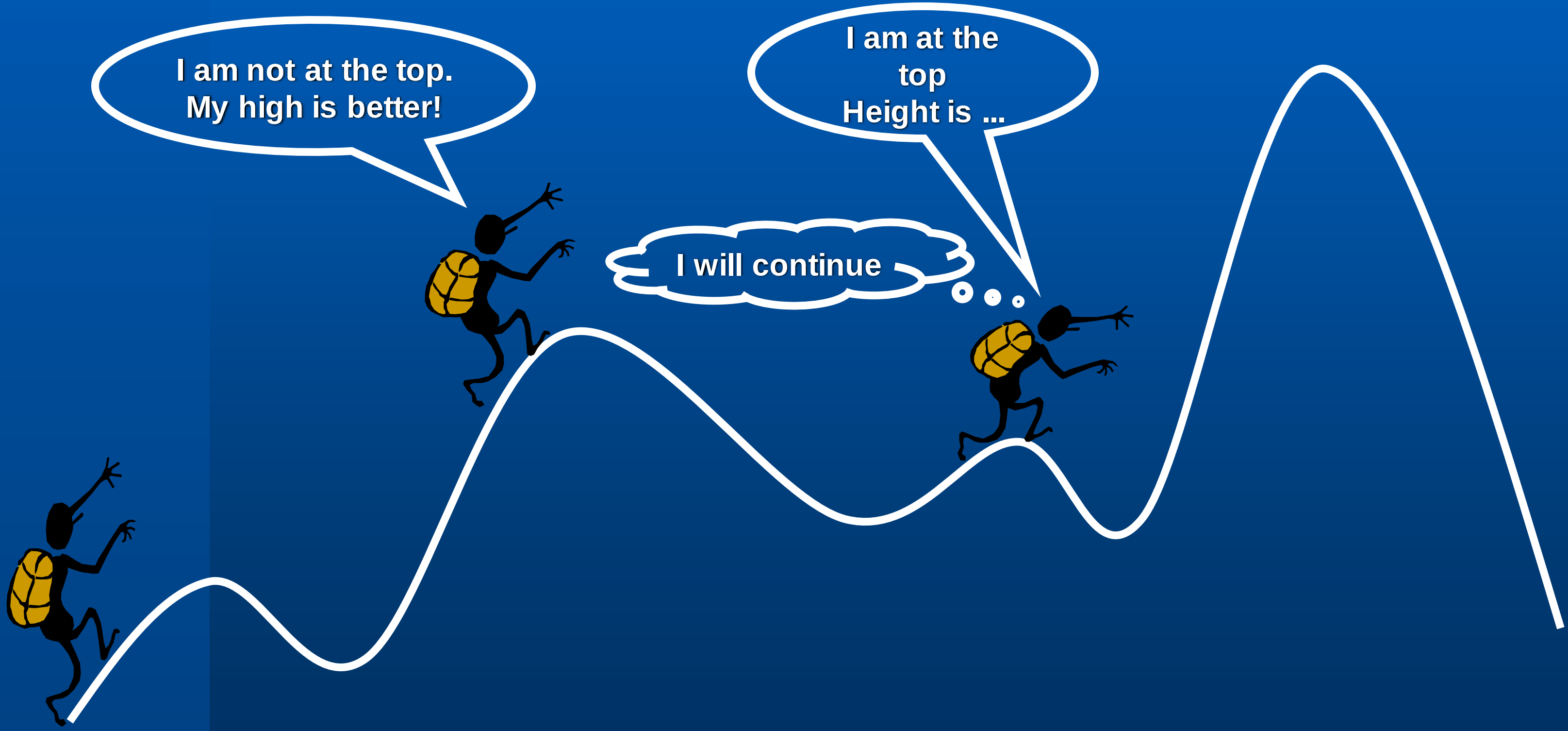


Instead of Introduction…(2)

- **Multi-climbers**

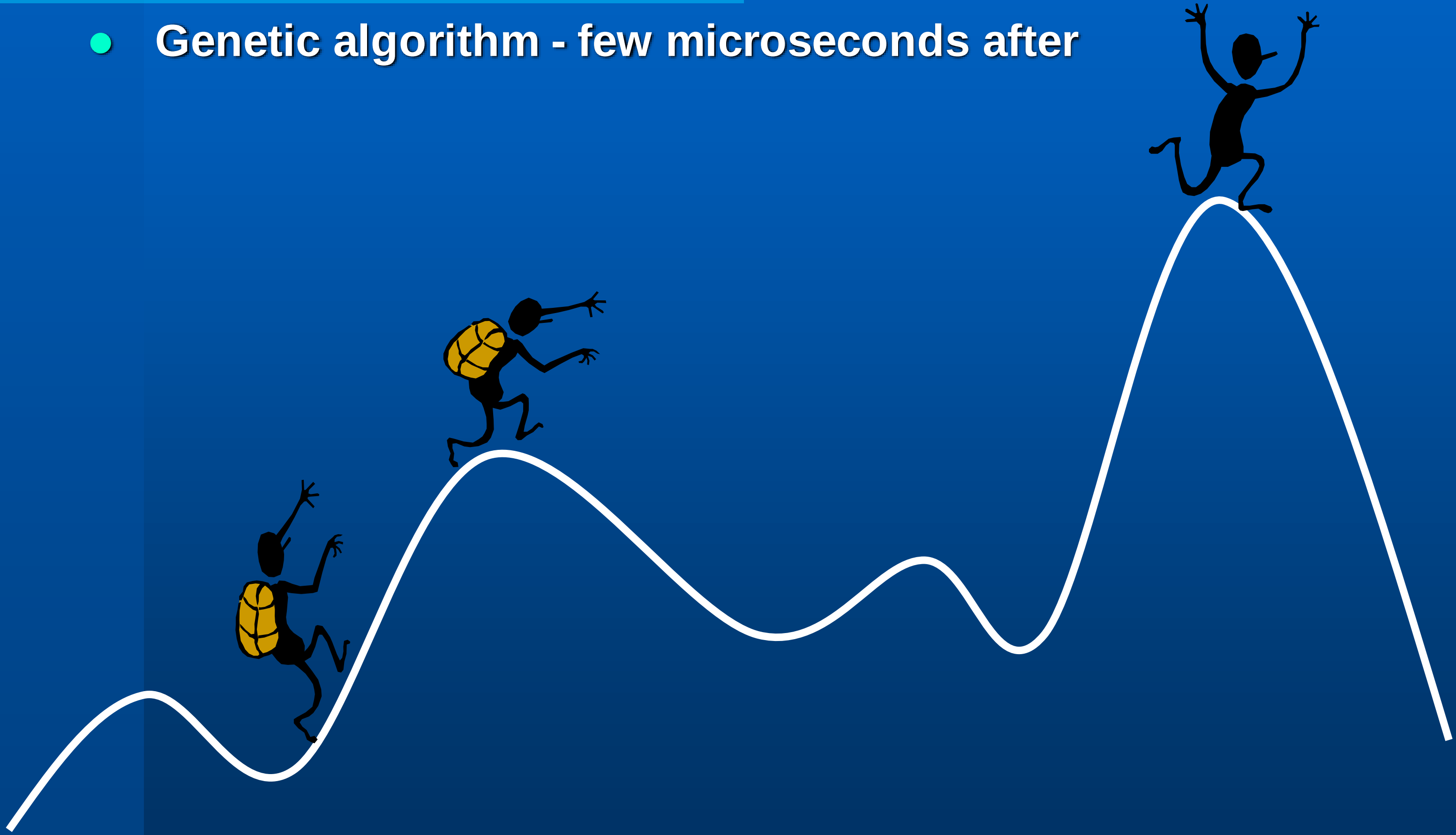
Instead of Introduction…(3)

- Genetic algorithm



Instead of Introduction…(3)

- **Genetic algorithm - few microseconds after**

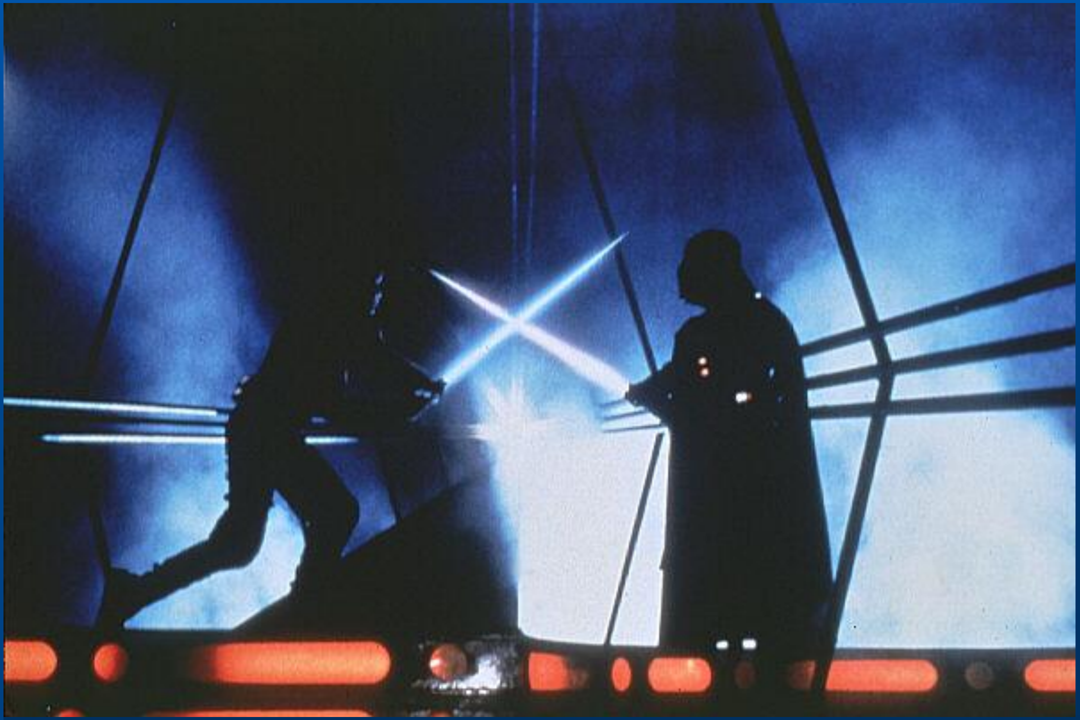


GA Concept

- Genetic algorithm (GA) introduces the principle of evolution and genetics into search among possible solutions to given problem.
- The idea is to simulate the process in natural systems.
- This is done by the creation within a machine of a population of individuals represented by chromosomes, in essence a set of character strings, that are analogous to the DNA, that we have in our own chromosomes.

Survival of the Fittest

- The main principle of evolution used in GA is “survival of the fittest”.
- The good solution survive, while bad ones die.



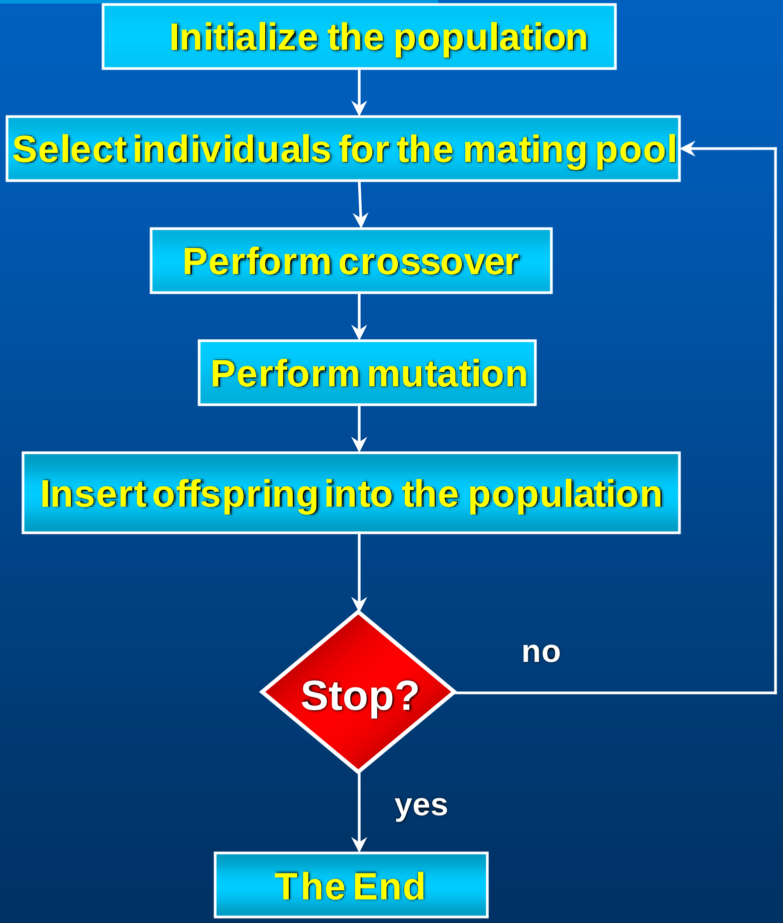
Nature and GA...

Nature	*Genetic algorithm*
Chromosome	String
Gene	Character
Locus	String position
Genotype	Population
Phenotype	Decoded structure

The History of GA

- Cellular automata
  - John Holland, university of Michigan, 1975.
- Until the early 80s, the concept was studied theoretically.
- In 80s, the first “real world” GAs were designed.

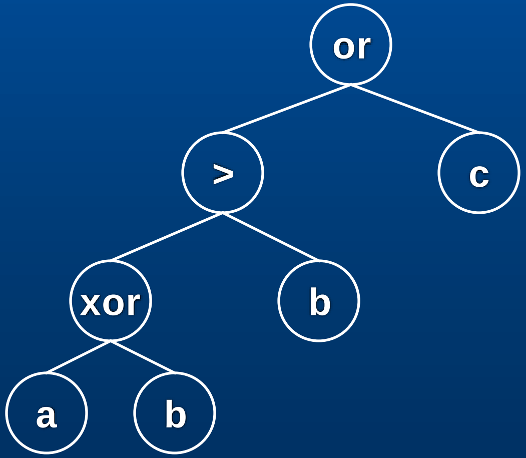
Algorithmic Phases


Initialize the population
Select individuals for the mating pool
Perform crossover
Perform mutation
Insert offspring into the population
Stop?
no
yes
The End

Designing GA...

- How to represent genomes?
- How to define the crossover operator?
- How to define the mutation operator?
- How to define fitness function?
- How to generate next generation?
- How to define stopping criteria?

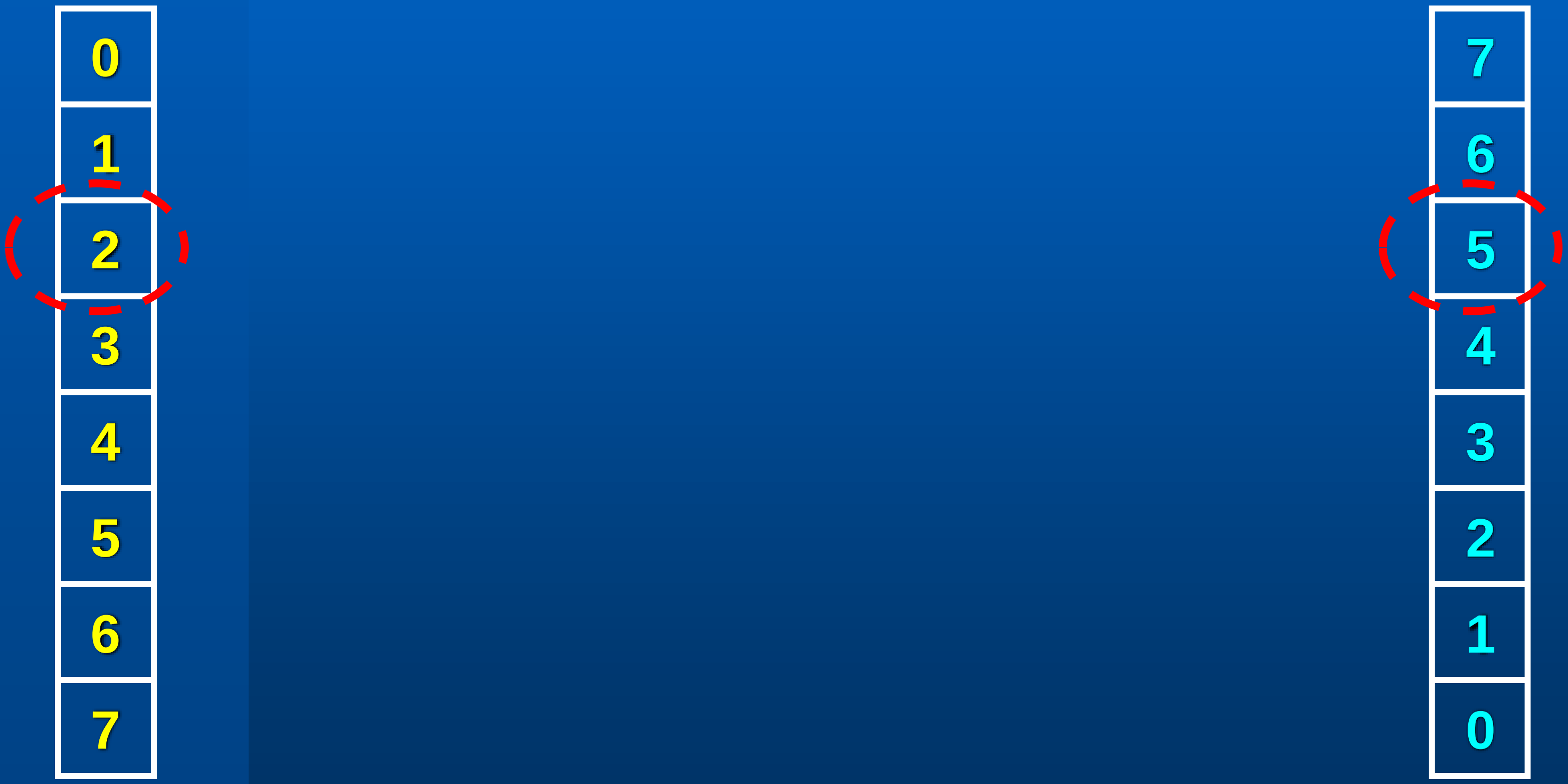
Representing Genomes...

Representation	Example
string	1 | 0 | 1 | 1 | 1 | 0 | 0 | 1
array of strings	http | avala | yubc | net | ~apopovic
tree - genetic programming	

Crossover

- Crossover is concept from genetics.
- Crossover is sexual reproduction.
- Crossover combines genetic material from two parents, in order to produce superior offspring.
- Few types of crossover:
  - One-point
  - Multiple point.

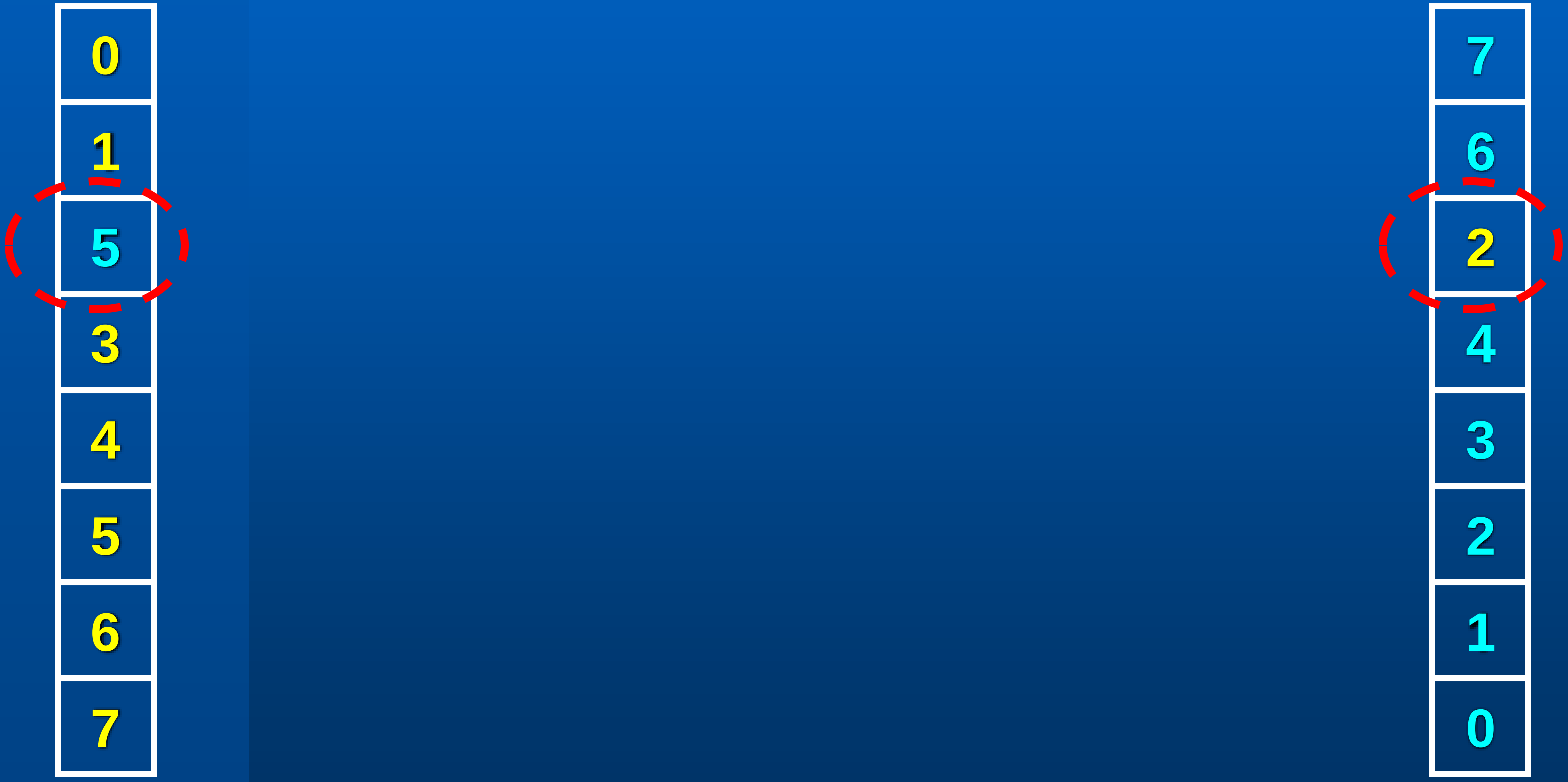
One-point Crossover



Parent #1

Parent #2

One-point Crossover



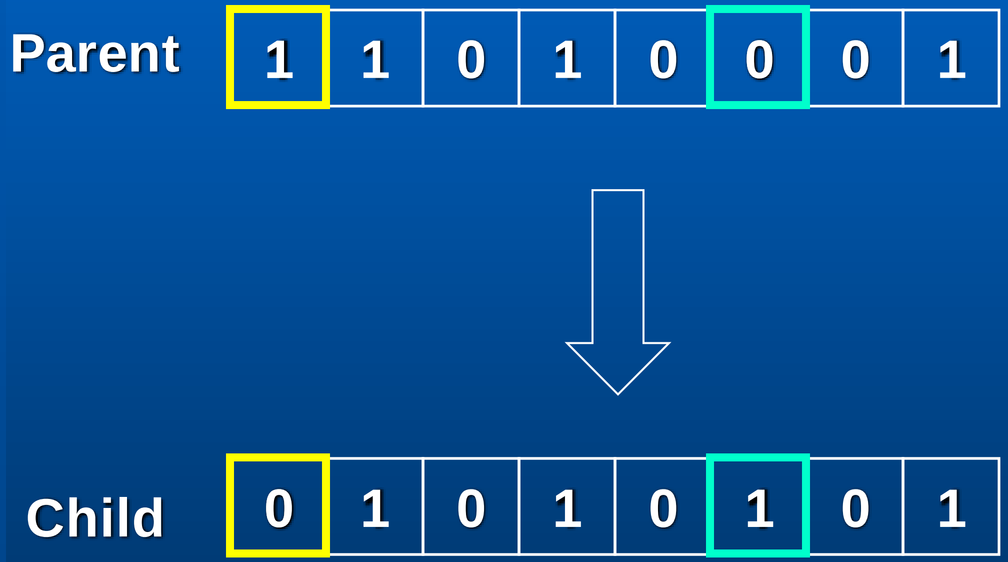
Parent #1

Parent #2

Mutation

- Mutation introduces randomness into the population.
- Mutation is asexual reproduction.
- The idea of mutation
  is to reintroduce divergence
  into a converging population.
- Mutation is performed
  on small part of population,
  in order to avoid entering unstable state.

Mutation...



About Probabilities...

- Average probability for individual to crossover is, in most cases, about 80%.
- Average probability for individual to mutate is about 1-2%.
- Probability of genetic operators follow the probability in natural systems.
- The better solutions reproduce more often.

Fitness Function

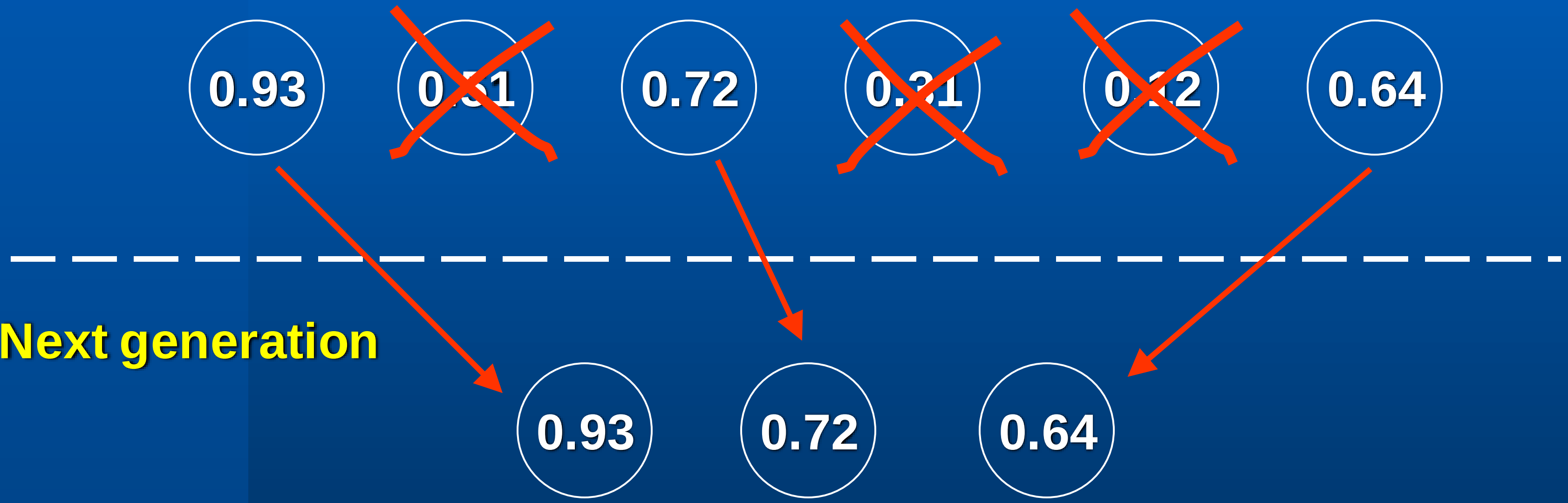
- Fitness function is evaluation function,
  that determines what solutions are better than others.
- Fitness is computed for each individual.
- Fitness function is application depended.

Selection

- The selection operation copies a single individual, probabilistically selected based on fitness, into the next generation of the population.
- There are few possible ways to implement selection:
  - "Only the strongest survive"
    - Choose the individuals with the highest fitness for next generation
  - "Some weak solutions survive"
    - Assign a probability that a particular individual will be selected for the next generation
    - More diversity
    - Some bad solutions might have good parts!

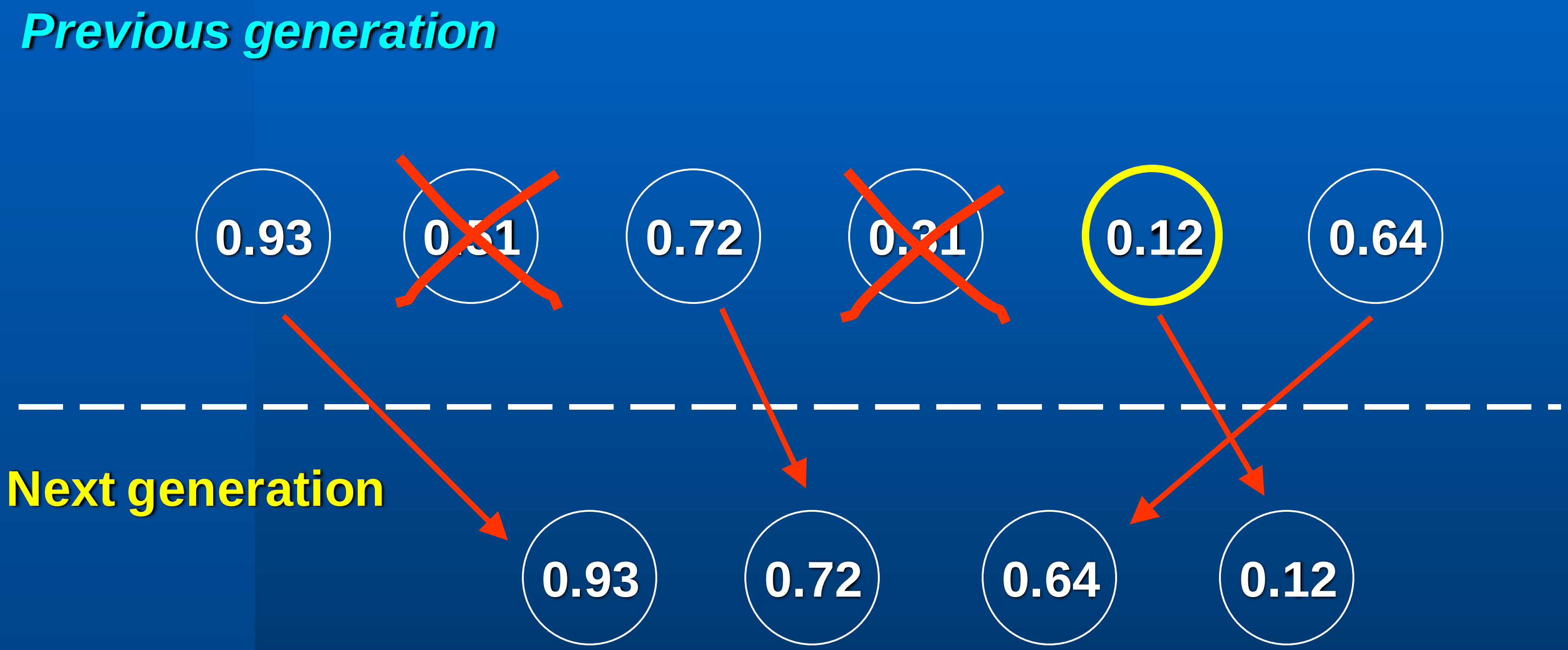
Selection - Survival of The Strongest

Previous generation

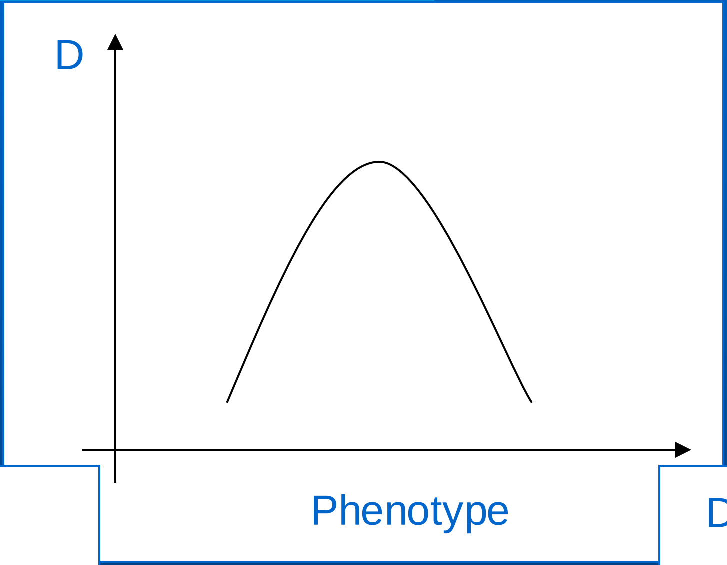


Next generation

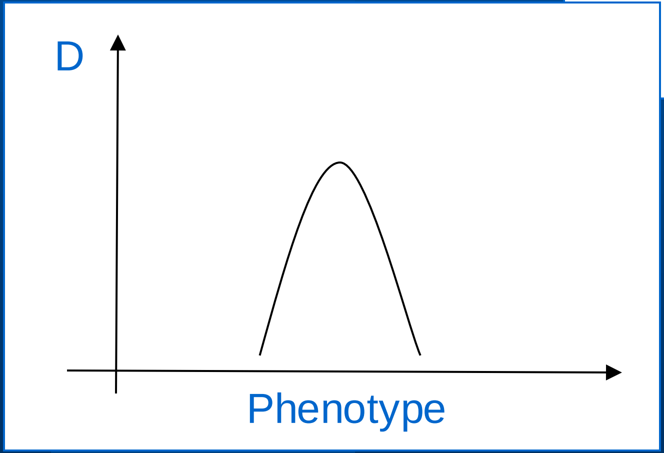
Selection - Some Weak Solutions Survive



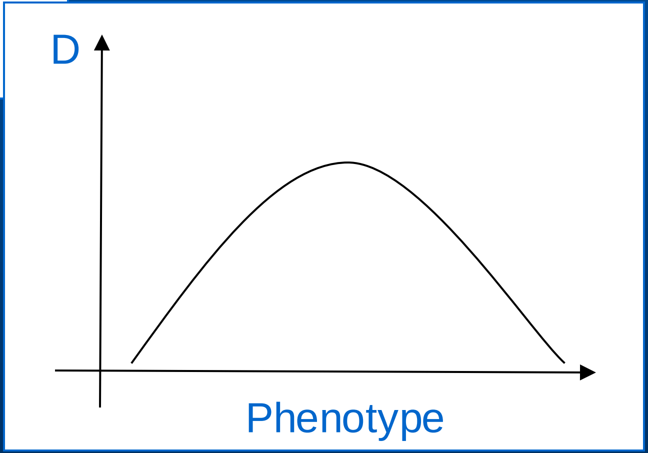
Mutation and Selection...



Solution distribution



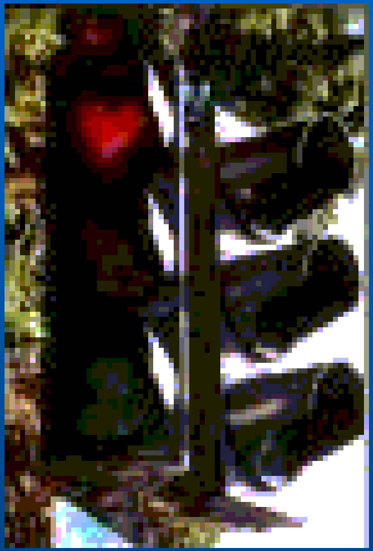
Selection



Mutation

Stopping Criteria

- Final problem is to decide when to stop execution of algorithm.
- There are two possible solutions to this problem:
  - First approach:
    - Stop after production of definite number of generations
  - Second approach:
    - Stop when the improvement in average fitness over two generations is below a threshold



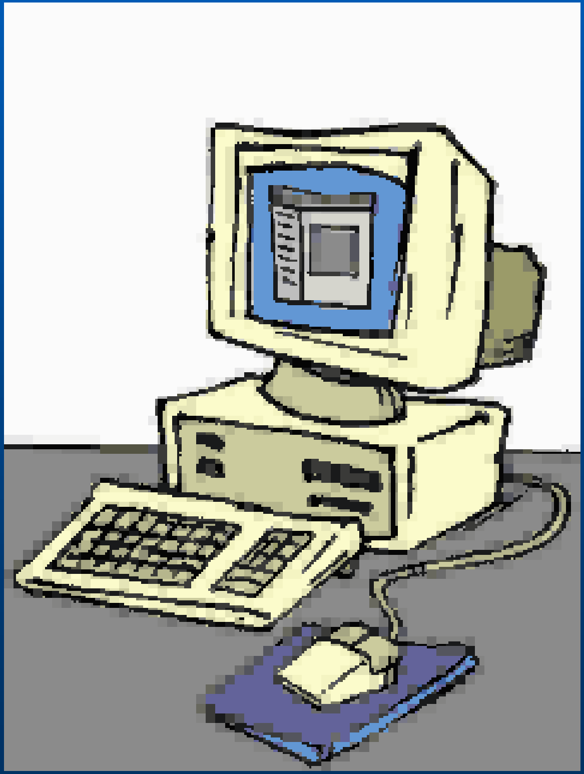
GA Vs. Ad-hoc Algorithms

		Genetic Algorithm	Ad-hoc Algorithms
✗	**Speed**	Slow *	Generally fast
✓	**Human work**	Minimal	Long and exhaustive
✓	**Applicability**	General	There are problems that cannot be solved analytically
✓	**Performance**	Excellent	Depends

*** Not necessary!**

Problems With GAs

- <u>Sometimes</u> GA is extremely slow,
  and much slower than usual algorithms



Advantages of GAs

- Concept is easy to understand.
- Minimum human involvement.
- Computer is not learned how to use existing solution, but to find new solution!
- Modular, separate from application
- Supports multi-objective optimization
- Always an answer; answer gets better with time !!!
- Inherently parallel; easily distributed
- Many ways to speed up and improve a GA-based application as knowledge about problem domain is gained
- Easy to exploit previous or alternate solutions

GA: An Example - Diophantine Equations

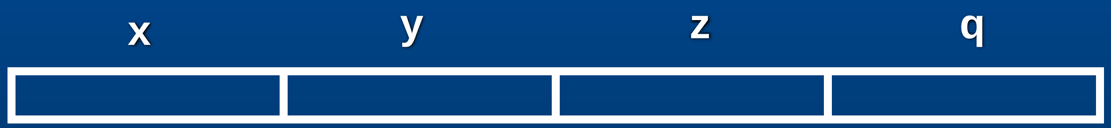
- Diophantine equation (n=4):

  A*x + b*y + c*z + d*q = s

- For given a, b, c, d, and s - find x, y, z, q

- Genome:

  (X, y, z, p) =



GA:An Example - Diophantine Equations(2)

- **Crossover**

$$\begin{tikzcd} (1, 2, 3, 4) \arrow[dd, leftrightarrow] & & (1, 6, 3, 4) \\ & {} \arrow[r, Rightarrow] & {} \\ (5, 6, 7, 8) & & (5, 2, 7, 8) \end{tikzcd}$$

- **Mutation**

$$(1, 2, 3, \underset{\uparrow}{4}) \quad \Longrightarrow \quad (1, 2, 3, \underset{\uparrow}{9})$$

GA:An Example - Diophantine Equations(3)

- First generation is randomly generated of numbers lower than sum (s).
- Fitness is defined as absolute value of difference between total and given sum:

  Fitness = abs (total - sum) ,

- Algorithm enters a loop in which operators are performed on genomes: crossover, mutation, selection.
- After number of generation a solution is reached.

Part II: Mathematics Behind GA-s

Two methods for analyzing genetics algorithms:

Schema analyses

Mathematical modeling

Schema Analyses

Weaknesses:

- In determining some characteristics of the population
- Schema analyses makes some approximations that weaken it

Advantages:

- A simple way to view the standard GA
- They have made possible proofs of some interesting theorems
- They provide a nice introduction to algorithmic analyses

Schema Analyses…

Schema – a template made up of a string of 1s, 0s, and *s,
where * is used as a wild card that can be either 1 or 0

For example, H = 1 * * 0 * 0 is a schema.

- It has eight instances (one of which is 101010)
- Order, o (H) , the number of non-*, or defined, bits (in example 3)
- Defining length, d (H) , greatest distance between two defined bits (in example H has a defining length of 5)

Let S be the set of all strings of length *l*.

- There is 3^l possible schemas on S, but 2^{2^l} different subsets of S
- Schema cannot be used to represent every possible population within S, but forms a representative subset of the set of all subsets of S

Schema analyses…

The end-of-iterations conditions:

expected number of instances of schema H as we iterate the GA

- **M(H, t) – the number of instances of H at time t**
- **f(x) – fitness of chromosome x**
- $\overline{f}(t)$ **- average fitness at time t :**

$$\overline{f}(t) = \frac{\sum_{x \in S} f(x)}{n}$$ **, n=|S|**

- $\hat{u}(H,t)$ **- average fitness of instances of H at time t:**

$$\hat{u}(H,t) = \frac{\sum_{x \in H} f(x)}{m(H,t)}$$

Schema Analyses…

- If we completely ignore the effects of crossover and mutation, we get the expected value:

$$E(m(H,t+1)) = n\frac{\sum_{x\in H} f(x)}{\sum_{x\in S} f(x)} = \frac{\sum_{x\in H} f(x)}{\bar{f}(t)} = \frac{\hat{u}(H,t)m(H,t)}{\bar{f}(t)}$$

- Now we consider only the effects of crossover and mutation, which lower the number of instances of H in the population. Then we will get a good lower bound on E(m(H,t+1))

- $S_c(H)$ - probability that a random crossover bit is between the defining bits of H
- p_c - probability of crossover occurring

$$S_c(H) = 1 - p_c\left(\frac{d(H)}{l-1}\right)$$

Schema Analyses…

- $S_m(H)$ - probability of an instance of H remaining the same after mutation; it is dependent on the order of H
- p_m - probability of mutation

$$S_m(H) = (1 - p_m)^{o(H)}$$

- With the above notation , we have:

Schema Theorem, provided by John Holland

$$E(m(H,t+1)) \geq \frac{\hat{u}(H,t)m(H,t)}{\bar{f}(t)}(1 - p_c \frac{d(H)}{l-1})((1 - p_m)^{o(H)})$$

Schema analyses…

⇨ The Schema Theorem only shows how schemas dynamically change, and how short, low-order schemas whose fitness remain above the average mean receive

exponentially growing increases in the number of samples.

⇨ It cannot make more direct predictions about the population composition, distribution of fitness and other statistics more directly related to the GA itself.

Mathematical Model

- One change is that one of the new individuals will be immediately deleted (thus the loop is iterated *n* times, not *n/2*)
- S – set of all strings of length l
- N – size of S, or 2^l
- $\vec{p}(t)$ column vector with 2^l rows such that the *i*-th component $\vec{p}_i(t)$ is equal to the proportion of the population P as time *t* that has chromosome *i*
- $\vec{s}(t)$ column vector with 2^l rows such that *i*- th component $\vec{s}_i(t)$ is equal to the probability that chromosome i will be selected as a parent

Mathematical Model…

Example: *l=2, 3* **individuals in the population, two with chromosome 10 and one with chromosome 11, then**

$$\vec{p}(t) = (0,0,\frac{2}{3},\frac{1}{3})^T$$

If the fitness is equal to the number of 1s in the string, then

$$\vec{s}(t) = (0,0,\frac{1}{2},\frac{1}{2})^T$$

Mathematical Model…

- $F = (F_{i,j})$ diagonal $N \times N$ matrix with $F_{i,i} = f(i)$
- Relation between $\vec{p}$ and $\vec{s}$: $\vec{s}(t) = \frac{F\vec{p}(t)}{|F\vec{p}(t)|}$
- Goal: given a column vector $\vec{x}$, to construct a column-vector-valued function $M(\vec{x})$ such that

$$M(\vec{s}(t)) = \vec{p}(t+1)$$

- *M* represents recombination – composition of crossover and mutation
- $i \oplus j$ component wise sum of i and j mod 2
- $i \otimes j$ component wise product of i and j
- $\mathrm{M} = (\mathrm{M}_{i,j})$ matrix whose i , j th entry $\mathrm{M}_{i,j}$ is the probability that 0 result from the recombination of i and j

Mathematical Model…

- σ_j permutation operator on R^N

$$\sigma_j((y_o,\ldots,y_{N-1})^T) = (y_{j\oplus 0},\ldots,y_{j\oplus(N-1)})^T$$

- Finally, $M(s)$ is given by the following expression

$$M(s) = ((\sigma_0 \vec{s}(t))^T \mathbf{M}_{\sigma_0} \vec{s}(t),\ldots,(\sigma_{2^l-1}\vec{s}(t))^T \mathbf{M}_{\sigma_{2^l-1}} \vec{s}(t))$$

- With this expression, we can calculate explicitly the expected value of each generation from the proceeding generation.

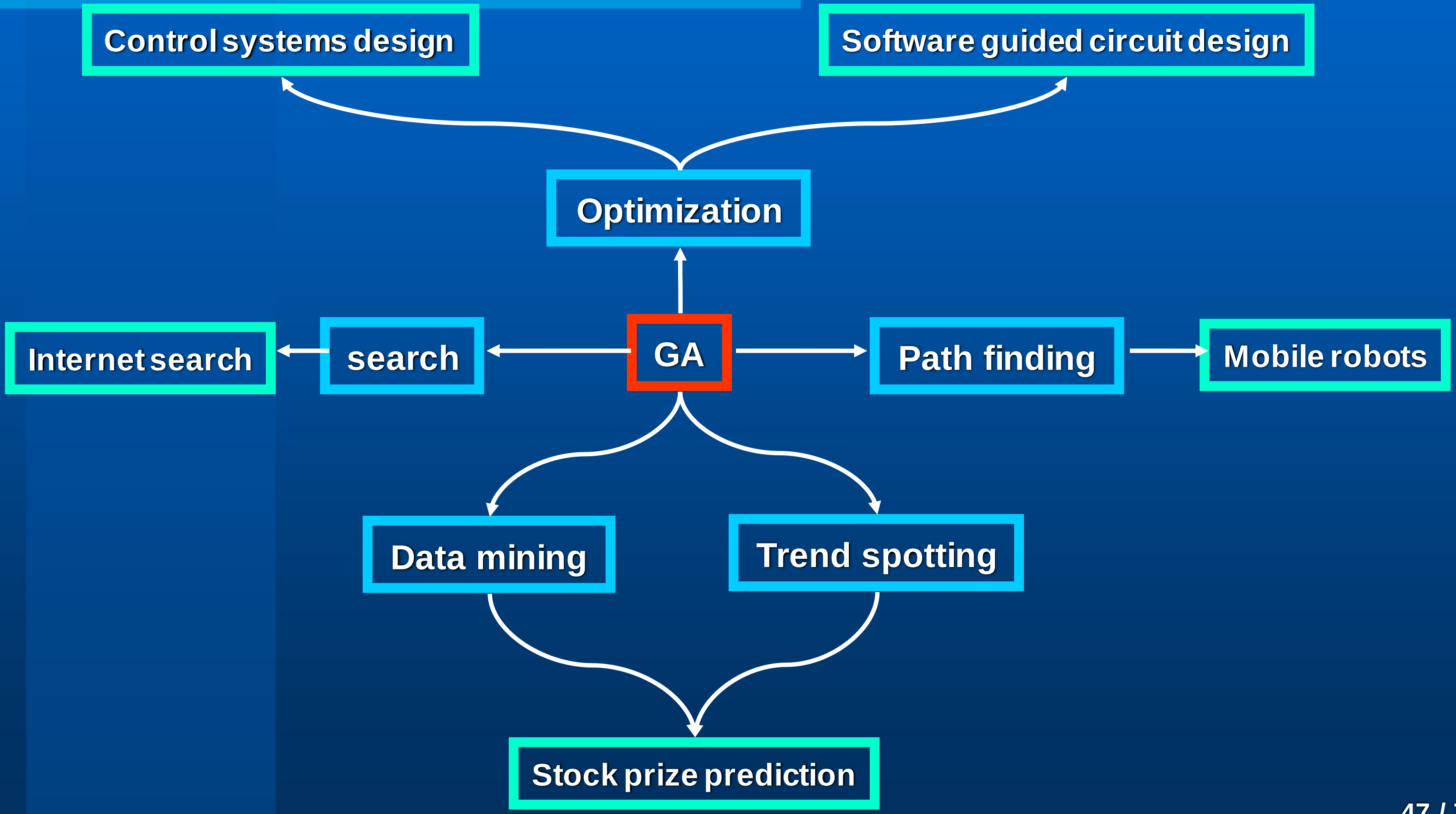
I hope you now fully understand the mathematics behind GA.

Part III: Applications of GAs

GA and the Internet

Genetic search based on multiple mutation approaches

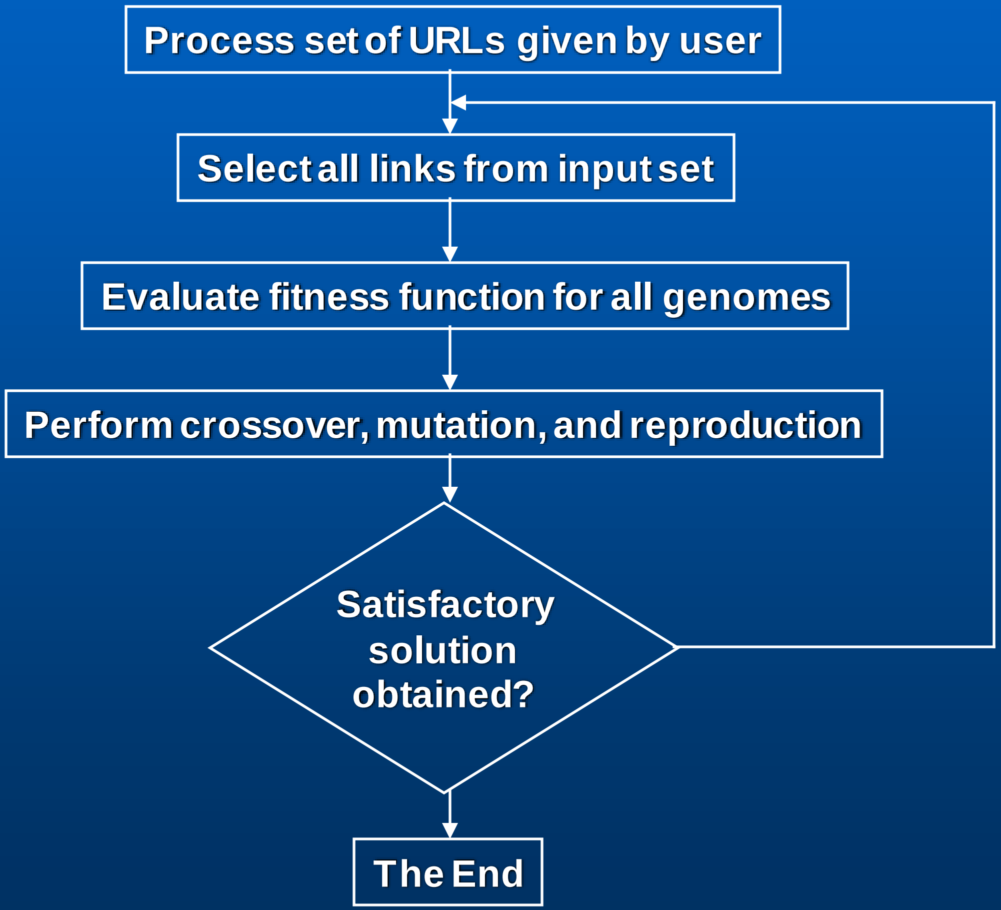
Some Applications of Gas



Genetic Algorithm and the Internet

The system designed by EBI Group, Faculty for Electrical Engineering, University of Belgrade

Algorithms Phases

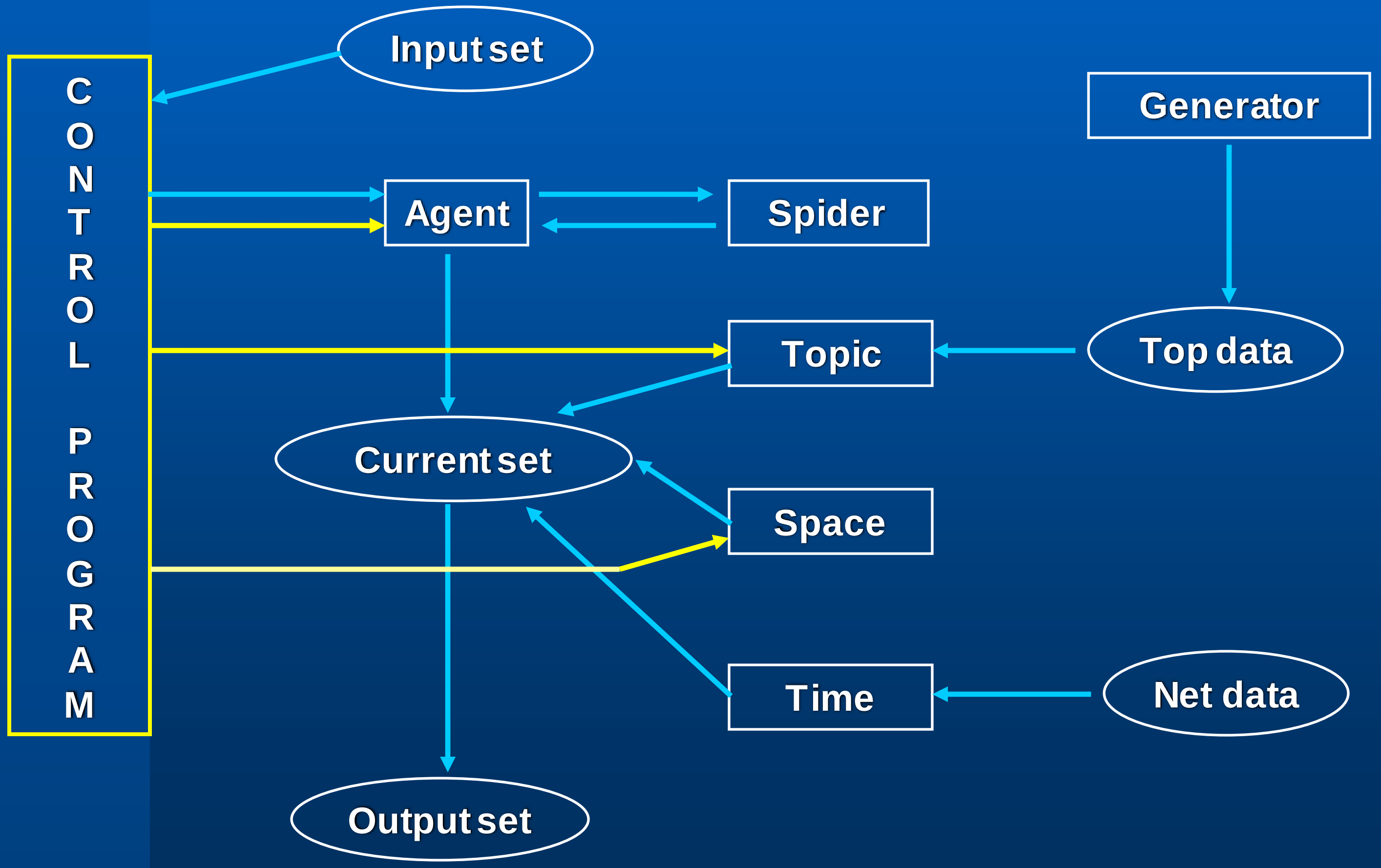

Process set of URLs given by user
Select all links from input set
Evaluate fitness function for all genomes
Perform crossover, mutation, and reproduction
Satisfactory solution obtained?
The End

Introduction

- GA can be used for intelligent internet search.
- GA is used in cases when search space is relatively large.
- GA is adoptive search.
- GA is heuristic search method.

System for GA Internet Search

- Designed at faculty for electrical engineering, university of belgrade



Spider

- Spider is software packages,
  that picks up internet documents
  from user supplied input with depth specified by user.
- Spider takes one URL, fetches all links,
  and documents thy contain with predefined depth.
- The fetched documents are stored on local hard disk with same structure as on the original location.
- Spider's task is to produce the first generation.
- Spider is used during crossover and mutation.

Agent

- Agent takes as an input a set of urls,
  and calls spider, for every one of them, with depth 1.
- Then, agent performs extraction of keywords
  from each document, and stores it in local hard disk.

Generator

- Generator generates a set of urls from given keywords, using some conventional search engine.
- It takes as input the desired topic, calls yahoo search engine, and submits a query looking for all documents covering the specific topic.
- Generator stores URL and topic of given web page in database called topdata.

Topic

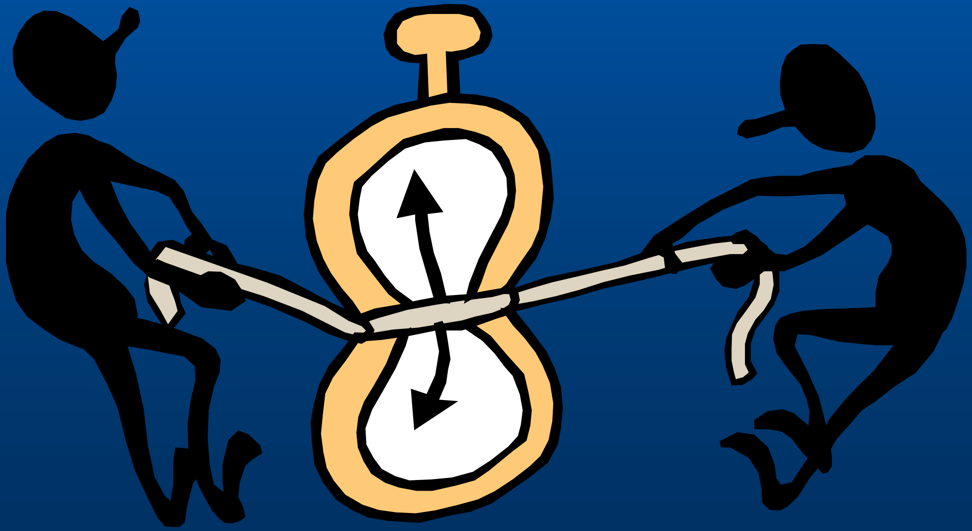
- It uses topdata DB in order to insert random urls from database into current set.
- Topic performs mutation.

Space

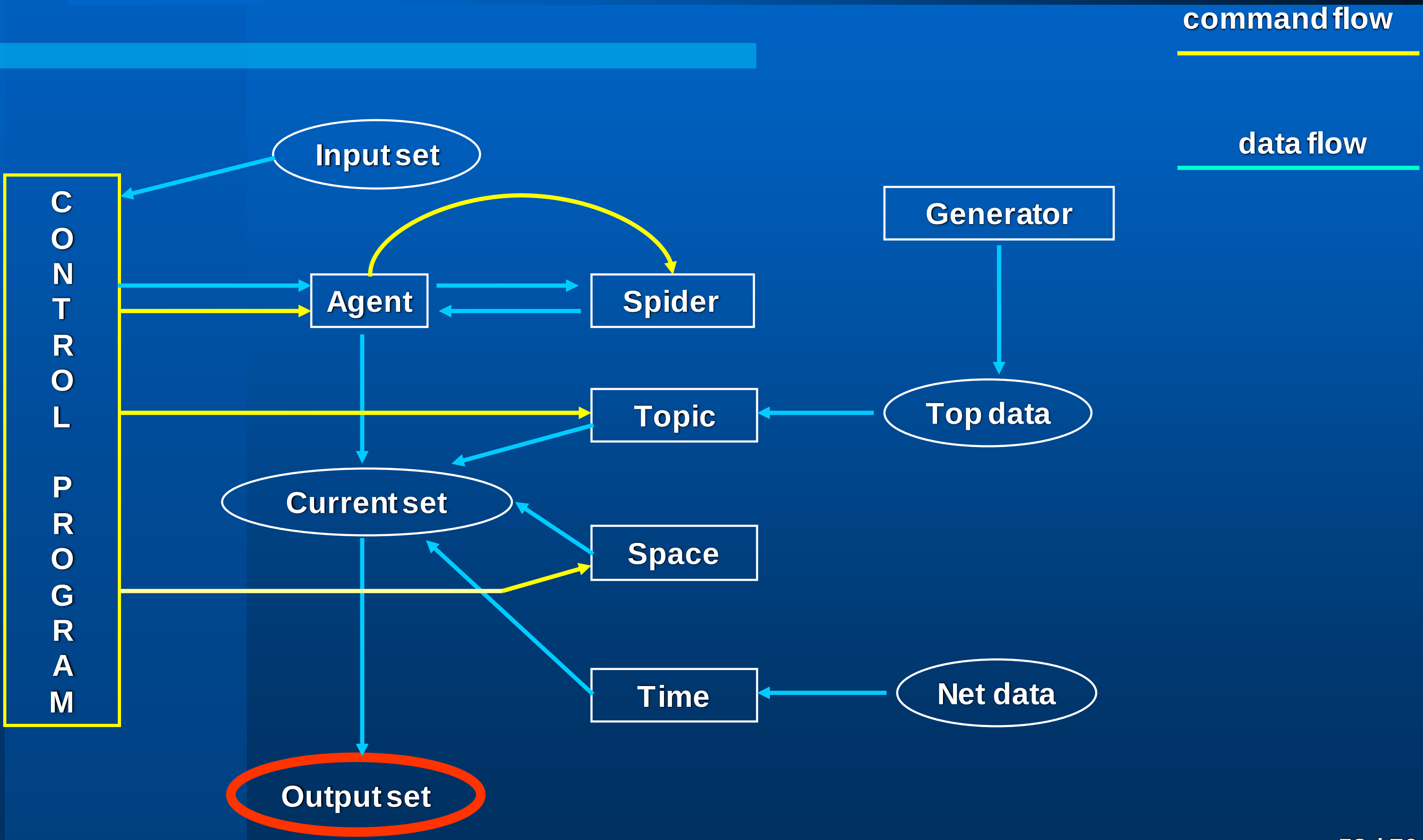
- Space takes as input the current set
  from the agent application
  and injects into it those urls
  from the database netdata
  that appeared with the greatest frequency
  in the output set of previous searches.

Time

- Time takes set of urls from agent
  and inserts ones with greatest frequency into DB netdata.
- The netdata DB contains of three fields: URL, topic,
  and count number.
- The DB is updated in each algorithm iteration.

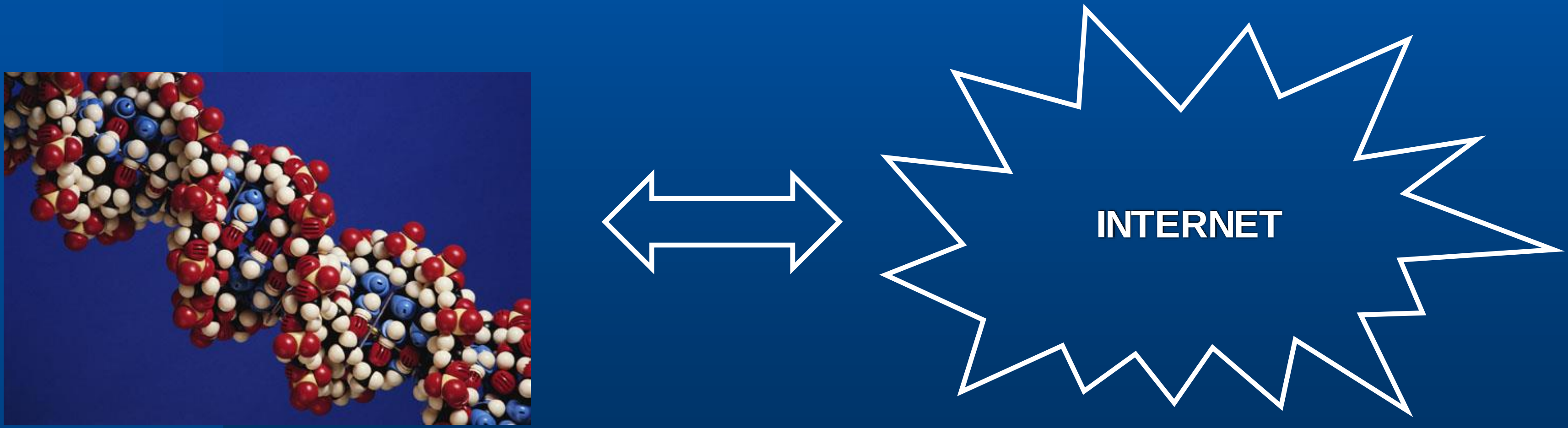


How Does The System Work?



GA and the Internet: Conclusion

- GA for internet search, on contrary to other gas, is much faster and more efficient that conventional solutions, such as standard internet search engines.



Genetic Search Based on Multiple Mutation Approaches

Concept and its improvements adapted to specific applications in e-business, and concrete software package

Main problems in finding information on the Internet:

- How to find quickly and retrieve efficiently the potentially useful information considering the fact of the fast growth of the quantity and variety of Internet sites
- Huge number of documents , many of which are completely unrelated to what the user originally attempted to find, searched with indexing engines
- Documents placed on the top of the result list are often less acceptable then the lower ones
- Indexing process may take days, weeks , or even longer, because the volume of new information being created daily

Links Based Approach

The question is:

How to locate and retrieve the needed information before it gets indexed?

The efficient way to locate the new not-yet-indexed information:

$$\begin{tikzcd}
\bullet\ \text{Using links-based approaches} \arrow[r] \arrow[dr] & \text{genetic search} \\
& \text{simulated annealing}
\end{tikzcd}$$

Best result:

indexing - based approaches

+

links - based approaches

Genetic Search Algorithm

GENETIC ALGORITHM OF ZERO ORDER, with no mutation

Start:

Model Web presentation that contains all the needed types of information (fitness function is evaluated).

It is assumes that it includes URL pointers to other similar Web presentations, and these are downloaded.

The Web presentations that "survived" the fitness function are assumed to include additional URL pointers, and their related Web presentations are downloaded next.

After the end-of-search condition is met, the Web presentations are ranked according to their fitness value.

Genetic Search Algorithm…

Type of mutation:

- Topic-oriented database mutation
- Semantic mutations
  - based on the principles of spatial locality
  - based on the principles of temporal locality

Logical reasoning and semantics consideration is involve in picking out URLs for mutation.

Innovations Required by Domain Area

- APPLICATION LEVEL
- LEVEL OF THE GENERAL PROJECT APPROACH AND PRODUCT ARCHITECTURE
- ALGORITHMIC LEVEL
- IMPLEMENTATION LEVEL

Application Level

- Statistical analysis and data mining has to be performed, in order to figure out the common and typical patterns of behavior and need
- The state-of-the-art of mutual referencing has to be determined
- The trends and asymptotic situations foreseen for the time of project finalization has to be determined

Level of the General Project Approach and Product Architecture

Decisions have to be made about the most important goals to be achieved:

- Maximizing the speed of search
- Maximizing the sophistication of search
- Maximizing specific effects of interest for a given institution or a customer
- Maximizing a combination of the above

Decision on this level affect the applicability of the final product / tool.

Algorithmic Level

Develop an efficient mutation algorithm of interest for the application

$$\begin{tikzcd} & \cdot \arrow[dl] \arrow[dr] & \\ \text{} & & \text{} \end{tikzcd}$$

in the direction of database architecture and design

in introducing the elements of semantic-based mutation

Semantics-based mutations are especially of interest for chaotic markets, typical of new markets in developed countries or traditional markets in under-developed countries.

Semantics-based Mutation

Mutation based on spatial localities

- **After a “fruitful” Web presentation is reached (using a tradicional algorithm with mutation), the site of the same Internet service provider is searched for other presentations on the same or similar topic**

Explanation :

In chaotic markets, it is very unlikely that service/product offers from the same small geographic area each other on their Web presentations

After a successful “side trip” based on spatial mutation, one continue with the traditional database mutation.

Semantics-based Mutation…

Mutation based on temporal localities

- **One comes back periodically to a Web presentation which was “fruitful” in the past**
- **One comes back periodically to other Web presentations developed by the author who created some “fruitful” Web presentations in the past**

Temporal mutation can use direct revisits or a number of indirect forms or revisit.

Implementation Level

- Utilization of novel technologies, for maximal performance and minimal implementation complexity

  Important for:

  - good flexibility

  - extendibility

  - reliability

  - availability

- Utilization of mobile platforms and mobile agents

Implementation Level...

- Static agents

  - one has to download megabytes of information

  - treat that information with a decision-making code of size measured in kilobytes

  - derive the final business related decision, which is binary in size (one bit: yes or no)

  A huge amount of data is transferred through the network in vain, because only a small percent of fetched documents will turn out to be useful

- Mobile agents

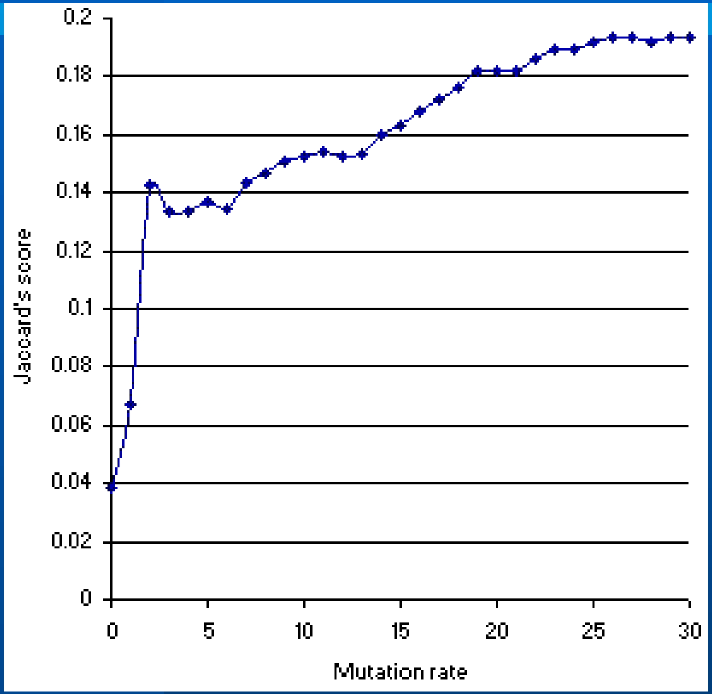
  - they would browse through the network and perform the search locally, on the remote servers, transferring only the needed documents and data

  - they load the network only with kilobytes and a single bit

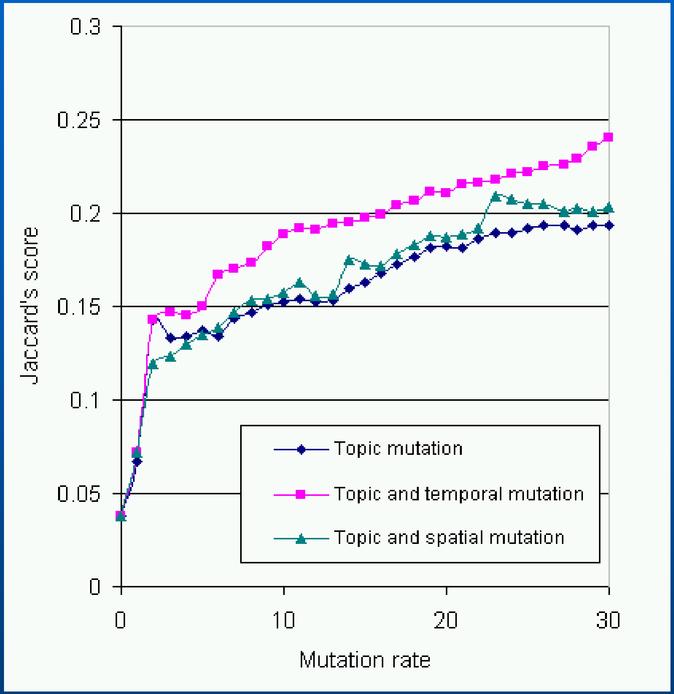
Simulation Result

- Links-based approach in the static domain
- How various mutation strategies can affect the search efficiency
- Set of software packages have developed , that would perform Internet search using genetic algorithms (by Veljko Milutinovic, Dragana Cvetkovic, and Jelena Mirkovic)
- As the fitness function they have measured average Jaccard’s score for the output documents, while changing the type and rate of mutation

Simulation Result…

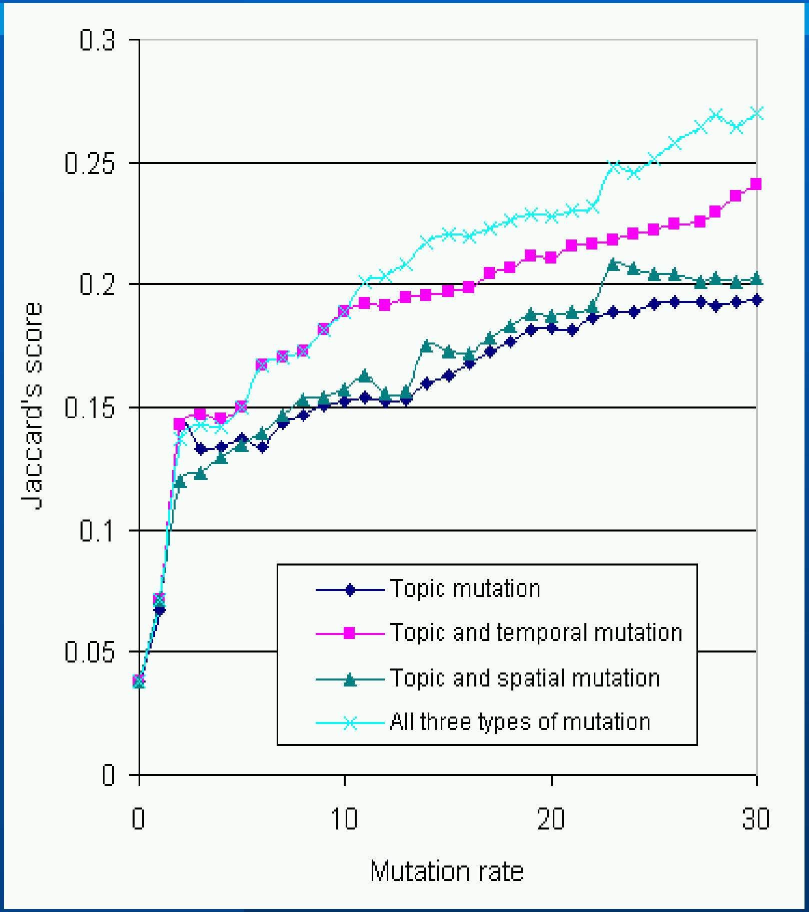


The simulation result for topic mutation



The simulation result for temporal and spatial mutation combined with topic mutation

Simulation Result…



The simulation result for topic, spatial and temporal mutation combined.

Constant increase in the quality of pages found.

Conclusion: Evolution

Tutorial download: galeb.etf.bg.ac.yu/~vm Option:Tutorials