Genetic Algorithms

The Fundamental Theorem

GA

Why does it work?

1. Schema Theorem
2. Search Spaces *as* Hypercubes

Schema

A **Schema** is a similarity template describing a subset of strings with similarities at certain string positions.

We can think of it as a pattern matching device: a schema matches a particular string if at every location in the schema a 1 matches a 1 in the string, or a 0 matches a 0, or a * matches either.

e.g. For a binary alphabet **{0, 1},** we motivate a schema by appending a special symbol *****, or **don't care symbol**,

producing a ternary alphabet **{0, 1, *}** that allows us to build schemata.

Notation: String, Population

Consider strings to be constructed over the binary alphabet
$V=\{0, 1\}$;

- **Strings** as capital letters
- **Individual characters** by lowercase letters subscripted by their position.

Example:

A = 0111000 may be represented symbolically as:
$A = a_1 a_2 a_3 a_4 a_5 a_6 a_7$

a_i represents a **gene** (binary feature or detector)
a_i **value** represents an **allele**

$A(t)$ represents a population of strings at time (or generation) t.

Notation: Schema

Consider a schema **H** taken from the **three-letter** alphabet:

V={ 0, 1, * };

*** asterisk is a <u>don’t care symbol</u> which matches either a 0 or a 1 at a particular position.**

Schema Matching

A **bit string** matches a particular **schemata** if that bit string can be constructed from the schemata by replacing the symbol with the appropriate bit value.

e.g.

H = *11*0**

String A = 0111000

String A is an example of the schema H because the string alleles a_i match schema positions h_i at the fixed positions 2, 3 and 5.

Schema Properties

Understanding the building blocks of future solutions

Schema Properties

Defining Length of Schema:
δ(H) – is the distance between the first and last specific string position

011*1**

Schema Defining Length:
δ(H) = 5-1 = 4

Order of Schema:
o(H) – is the number of fixed positions present in the template

Schema Order:
o(011*1**) = 4

0******

Schema Defining Length: δ(H) = 0, because there is only one fixed position

Schema Order:
o(0******) = 1

Understanding the building blocks of future solutions

Schema Properties

Schemata and their properties serve as notational devices for rigorously discussing and classifying string similarities.

They provide the basic means for analyzing the **net effect of reproduction and genetic operators** on the building blocks contained within the population.

Growth and Decay of Schemata

Effect of Reproduction on Schemata

Suppose at time **t**, there are **m** examples of a particular schema **H** in population **A(t)**

$$m = m(H, t)$$

During reproduction, a string **A_i** gets copied according to its fitness with probability **p_i** = $\frac{f_i}{\sum f}$

After picking a non-overlapping population of size **n** with replacement from the population **A(t)**, we may write the **reproductive schema growth equation** as:

$$m(H, t+1) = m(H, t) * n * \frac{f(H)}{\sum f_j}$$

f(H) is the **average fitness** of the strings representing schema **H** at time **t**.

Effect of Reproduction on Schemata

After picking a non-overlapping population of size **n** with replacement from the population **A(t)**, we may write the reproductive schema growth equation as:

$$m(H,t+1) = m(H,t) * n * \frac{f(H)}{\sum f_j}$$

Simplification

If we recognise that the average fitness of the entire population as $\bar{f} = \frac{\sum f}{n}$ we may express the

reproductive schema growth equation as:

$$m(H,t+1) = m(H,t) * \frac{f(H)}{\bar{f}}$$

Effect of Reproduction on Schemata

Reproductive schema growth equation:

$$m(H,t+1) = m(H,t) * \frac{f(H)}{\bar{f}}$$

• A particular schema grows as the ratio of the average fitness of the schema to the average fitness of the population.

• Schemata with **fitness values** **above** the population average will receive an **increasing** number of samples in the next generation.

• Schemata with **fitness values** **below** the population average will receive a **decreasing** number of samples.

• **All the schemata** in a population grow or decay according to their schema averages under the operation of reproduction alone.

Quantitave Effect of Reproduction on Schemata

Reproductive schema growth equation:

$$m(H,t+1) = m(H,t) * \frac{f(H)}{\bar{f}}$$

Suppose we assume that a particular schema **H** remains above average an amount $c\bar{f}$ with a **c** constant. Under this assumption, we can write:

$$m(H,t+1) = m(H,t) * \frac{(\bar{f} + c\bar{f})}{\bar{f}} = (1+c) * m(H,t)$$

Starting at **t=0**, and assuming a stationary value of **c**, we obtain the equation:

$$m(H,t) = m(H,0) * (1+c)^t$$

Reproduction allocates exponentially increasing (decreasing) numbers of trials to above (below) average schema.

Quantitave Effect of Reproduction on Schemata

$$m(H,t) = m(H,0) * (1+c)^t$$

Reproduction can allocate exponentially increasing and decreasing numbers of schemata to future generations in parallel.

Many, many different schemata are sampled in **parallel** according to the same rule through the use of **n** simple reproduction operations.

However, **reproduction** **does not promote exploration of new regions of the search space**.

This is where **crossover** steps in.

Effect of Crossover on Schemata

Consider a particular string of length $\ell = 7$ and two representative schemata within that string:

$A = 0111000$
$H_1 = *1****0$
$H_2 = ***10**$

Recall: Crossover Operation

crossover proceeds with the random selection of a mate;
Random selection of a crossover site, and the exchange of substrings from the beginning of the string to the crossover site inclusively with the corresponding substring of the chosen mate.

Effect of Crossover on Schemata

Consider a particular string of length **ℓ = 7** and two representative schemata within that string:

```
A   = 0 1 1 1 0 0 0
H1 =  * 1 *  * * * 0
H2 =  * * *  1 0 * *
```

Assuming that we have the following randomly chosen **crossover site**: **3**

```
A  = 0 1 1 | 1 0 0 0
H1 = * 1 * | * * * 0
H2 = * * * | 1 0 * *
```

Effect of Crossover on Schemata

Assuming that we have the following randomly chosen **crossover site**: **3**

```
A  = 0 1 1 | 1 0 0 0
H1 = * 1 * | * * * 0
H2 = * * * | 1 0 * *
```

H_1 is destroyed. Defining length = 5
H_2 will survive. Defining length = 1

H_1 is less likely to survive crossover than schema **H_2** because on average the crossover site is more likely to fall between the extreme fixed positions.

Effect of Crossover on Schemata

A = 0 1 1 | 1 0 0 0
H_1 = * 1 * | * * * 0
H_2 = * * * | 1 0 * *

H_1 is destroyed. Defining length = 5
H_2 will survive. Defining length = 1

H_1 is less likely to survive crossover than schema **H_2** because on average the crossover site is more likely to fall between the extreme fixed positions.

Let's quantify this observation. If the **crossover site** is selected **uniformly** at random among the **ℓ-1**=7-1 = **6 possible sites**, then **H_1** is destroyed with probability **p_d** and survives with probability **p_s**.

H_1

$$p_d = \frac{\delta(H)}{(l-1)} = \frac{5}{6}$$

$$p_s = 1 - p_d = \frac{1}{6}$$

Effect of Crossover on Schemata

$A = 0\ 1\ 1\ |\ 1\ 0\ 0\ 0$

$H_1 = *\ 1\ *\ |\ *\ *\ *\ 0$

$H_2 = *\ *\ *\ |\ 1\ 0\ *\ *$

H_1 is destroyed. Defining length = 5
H_2 will survive. Defining length = 1

H_1 is less likely to survive crossover than schema **H_2** because on average the crossover site is more likely to fall between the extreme fixed positions.

If the **crossover site** is selected **uniformly** at random among the **ℓ-1**=7-1 = **6 possible sites.** Similarly, you can calculate the probability of destruction and survival for **H_2** as follows:

H_2

$$p_d = \frac{\delta(H)}{(l-1)} = \frac{1}{6}$$

$$p_s = 1 - p_d = \frac{5}{6}$$

Lower Bound on Crossover Survival Probability

To generalise, a schema survives when the cross over site falls outside the defining length. The survival probability under simple crossover is p_s

Lower Bound on Crossover Survival Probability

$$p_s = \frac{1 - \delta(H)}{(l - 1)}$$

Lower Bound on Crossover Survival Probability

To generalise, a schema survives when the cross over site falls outside the defining length. The survival probability under simple crossover is p_s

$$p_s = \frac{1-\delta(H)}{(l-1)}$$

If we consider the probability of performing a crossover operation to be p_c,

$$p_s \geq 1 - p_c \cdot \frac{\delta(H)}{(l-1)}$$

Combined Effect of Reproduction and Crossover

Assuming independence of the **reproduction** and **crossover** operations,

$$m(H,t+1) \geq m(H,t) * \frac{f(H)}{\bar{f}}\left[1 - P_c \frac{\delta(H)}{l-1}\right]$$

Combined Effect of Reproduction and Crossover

Assuming independence of the **reproduction** and **crossover** operations,

$$m(H,t+1) \geq m(H,t) * \frac{f(H)}{\bar{f}}\left[1 - P_c \frac{\delta(H)}{l-1}\right]$$

Schema H grows or decays depending upon a multiplication factor.

Combined Effect of Reproduction and Crossover

Assuming independence of the **reproduction** and **crossover** operations,

$$m(H,t+1) \geq m(H,t) * \frac{f(H)}{\bar{f}} \left[1 - P_c \frac{\delta(H)}{l-1} \right]$$

Schema H grows or decays depending upon a multiplication factor.

That factor depends on 2 things:

- whether the schema is above or below the population average
- whether the schema has relatively short or long defining length.

Combined Effect of Reproduction and Crossover

Assuming independence of the **reproduction** and **crossover** operations,

$$m(H,t+1) \geq m(H,t) * \frac{f(H)}{\bar{f}}\left[1 - P_c \frac{\delta(H)}{l-1}\right]$$

Schema H grows or decays depending upon a multiplication factor.

That factor depends on 2 things:

- whether the schema is above or below the population average
- whether the schema has relatively short or long defining length.ar

Clearly, those schemata with both above-average observed performance and short-defining lengths are going to be sampled at exponentially increasing rates.

Effect of Mutation

Mutation is the random alteration of a single position with probability $\mathbf{p_m}$

Effect of Mutation

Mutation is the random alteration of a single position with probability $\mathbf{p_m}$

In order for a schema **H** to survive, **all** of the specified positions must themselves survive.

A **single allele** survives with a **probability** $(1-\mathbf{p_m})$

Effect of Mutation

Mutation is the random alteration of a single position with probability p_m

In order for a schema **H** to survive, **all** of the specified positions must themselves survive.

A **single allele** survives with a **probability** $(1-p_m)$

Since each of the mutations is **statistically independent**, a particular schema **H** survives when each of the **o(H)** **fixed positions** within the schema survives.

Effect of Mutation

Mutation is the random alteration of a single position with probability $\mathbf{p_m}$

In order for a schema **H** to survive, **all** of the specified positions must themselves survive.

A **single allele** survives with a **probability** (1-$\mathbf{p_m}$)

Since each of the mutations is **statistically independent**, a particular schema **H** survives when each of the **o(H)** **fixed positions** within the schema survives.

The survival probability is multiplied by itself o(H) times:

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

Effect of Mutation

Since each of the mutations is **statistically independent**, a particular schema **H** survives when each of the **o(H) fixed positions** within the schema survives.

The survival probability is multiplied by itself o(H) times:

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

For small values of $\mathbf{p_m}$ ($\mathbf{p_m} \ll 1$), we can write:

$$1-o(H)\cdot p_m$$

Schema Theorem

Fundamental Theorem of Genetic Algorithms

Number of **H** Schema at time *t*

Schema average

Schema Defining Length

$$m(H,t+1) \geq m(H,t) * \frac{f(H)}{\bar{f}} \left[1 - P_c \frac{\delta(H)}{l-1} - o(H) P_m \right]$$

Expected Count of Schema **H** at time *(t+1)*

Population average

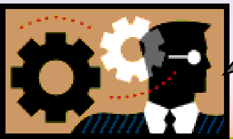
Schema Order

P_c – probability of cross over

P_m – probability of mutation

Who shall live and who shall die?

Short, low-order, above-average schemata are given exponentially increasing trials in subsequent generations.



Let's verify this!

How GA processes Schemata

Schema Processing by hand

Let us observe how the GA processes schemata-not individual strings-within the population.

Let us consider three particular schemata, H_1, H_2 and H_3

Where

- H_1 = 1****
- H_2 = *10**
- H_3 = 1***0

Observe the effect of reproduction, crossover, and mutation.

Genetic Algorithm

GA PROCESSING OF SCHEMATA – Hand Calculations

String number	Initial Population	X value	f(x)	pselect $\frac{f_i}{\sum f}$	Expected count $\frac{f_i}{\bar{f}}$	Actual count(Roulette Wheel)
1	01101	13	169	0.14	0.58	1
2	11000	24	576	0.49	1.97	2
3	01000	8	64	0.06	0.22	0
4	10011	19	361	0.31	1.23	1

Sum	1170
Ave.	293
Max.	576

Schema Processing: **Before Reproduction**

	Schema		String Representatives	Schema Average Fitness *f*(H)		
H_1	1****		2, 4	469		
H_2	*10**		2, 3	320		
H_3	1***0		2	576		

$$m(H,t+1) \geq m(H,t) * \frac{f(H)}{\bar{f}}\left[1 - P_c \frac{\delta(H)}{l-1} - o(H)P_m\right]$$

Genetic Algorithm

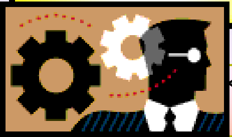
GA PROCESSING OF SCHEMATA – Hand Calculations

String number	Mating Pool after Reproduction	Mate (randomly selected)	Crossover site (random)	New population	X-value	$f(x)=x^2$
1	0110|1	2	4	01100	12	144
2	1100|0	1	4	11001	25	625
3	11|000	4	2	11011	27	729
4	10|011	3	2	10000	16	256

$$m(H,t+1) \geq m(H,t) * \frac{f(H)}{\bar{f}}\left[1-P_c\frac{\delta(H)}{l-1}-o(H)P_m\right]$$

Sum	1754
Ave.	439
Max.	729

After Reproduction			After All Operators			
Expected Count	Actual Count	String Representatives		Expected Count	Actual Count	String Representatives
3.20	3	2, 3, 4		3.20	3	2, 3, 4
2.18	2	2, 3		1.64	2	2, 3
1.97	2	2, 3		0.0	1	4



H_2 survives crossover with high probability. H_3 was reduced; a schema with this defining length is usually destroyed by crossover.

Search Spaces as Hyercubes

GA

Why does it work?

Search Spaces *as* Hypercubes

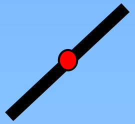
The question that most people who are new to the field of genetic algorithms ask at this point is why such a process should do anything useful?

Why should one believe that this is going to result in an effective form of search or optimization?

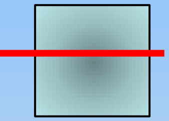
The answer which is most widely given to explain the **computational behavior of genetic algorithms** came out of John Holland's work. In his classic book, *Adaptation in Natural and Artificial Systems*, Holland develops several arguments designed to explain how a genetic plan or genetic algorithm can result in complex and robust search **by implicitly sampling hyperplane partitions of a search space**.

Hyperplane

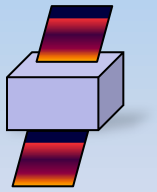
A **hyperplane** is a concept in geometry. It is a generalization of the concept of a plane.



- In **1-D** space (such as a line), **a hyperplane is a point**; it divides a line into two rays.



- In **2-D** space (such as the xy plane), **a hyperplane is a line**; it divides the plane into two half-planes.

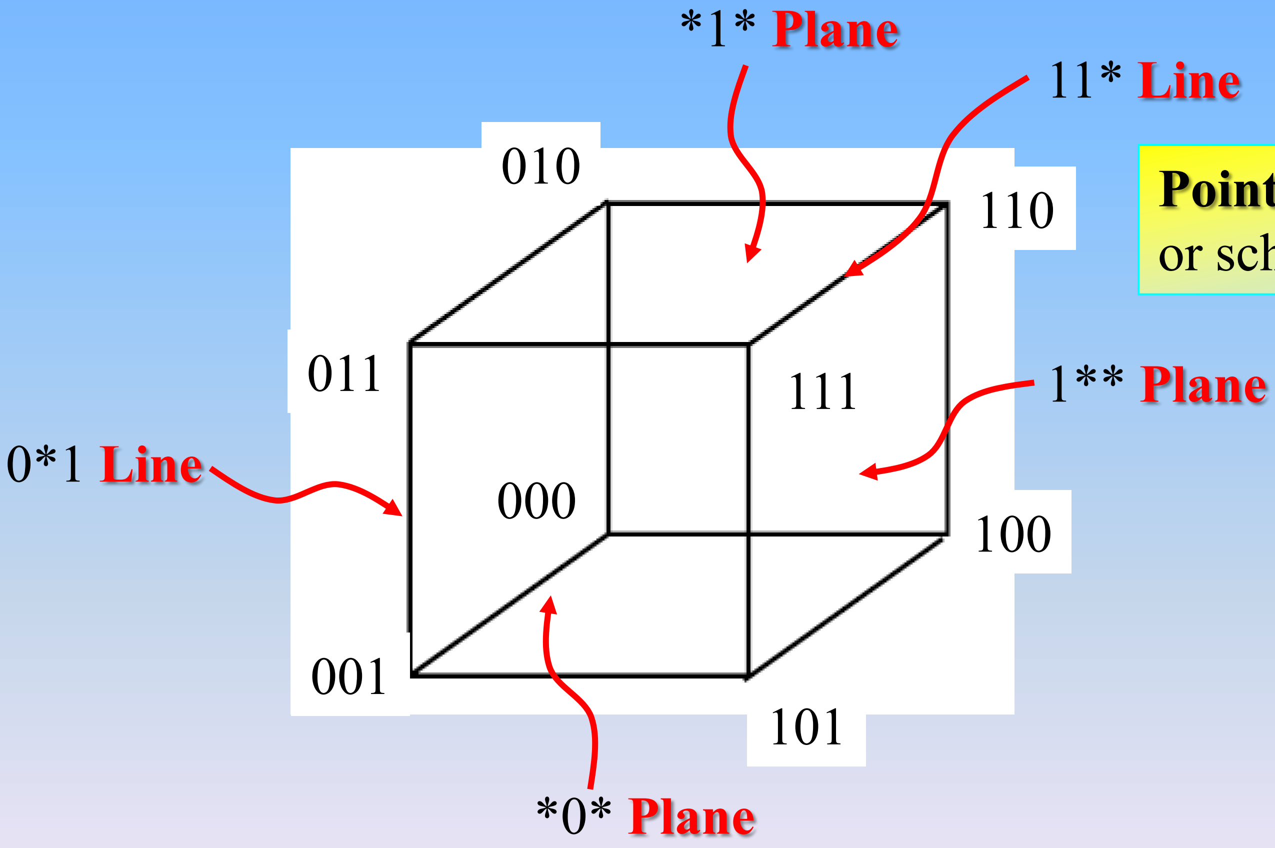


- In **3-D** space, **a hyperplane is an ordinary plane**; it divides the space into two half-spaces.
- This concept can also be applied to four-dimensional space and beyond, where **the dividing object is simply referred to as a hyperplane**.

http://en.wikipedia.org/wiki/Hyperplane

Visualization of Schemata as Hyperplanes in 3-D Space

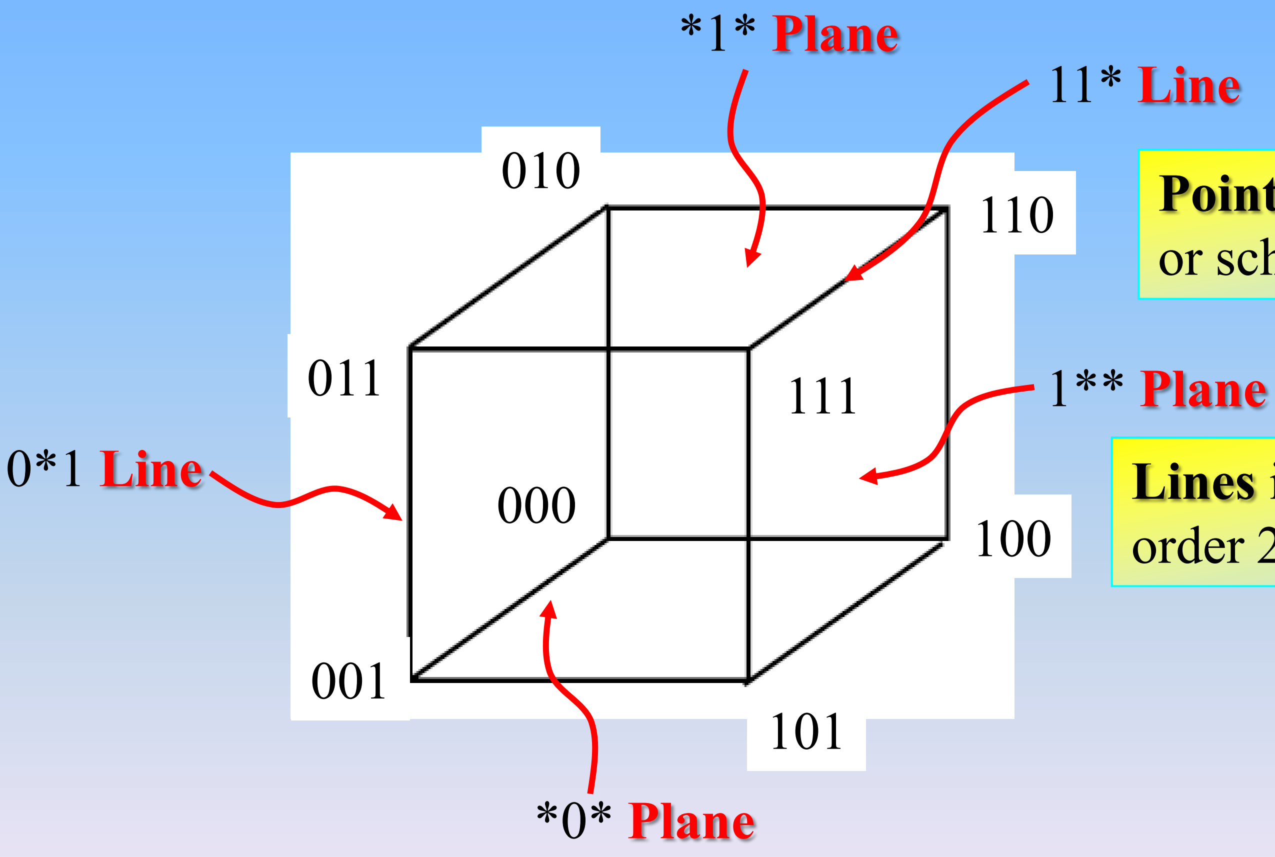
Consider the strings and schemata of length $\ell=3$



Points in the space are strings or schemata of order 3

Visualization of Schemata as Hyperplanes in 3-D Space

Consider the strings and schemata of length $\ell = 3$

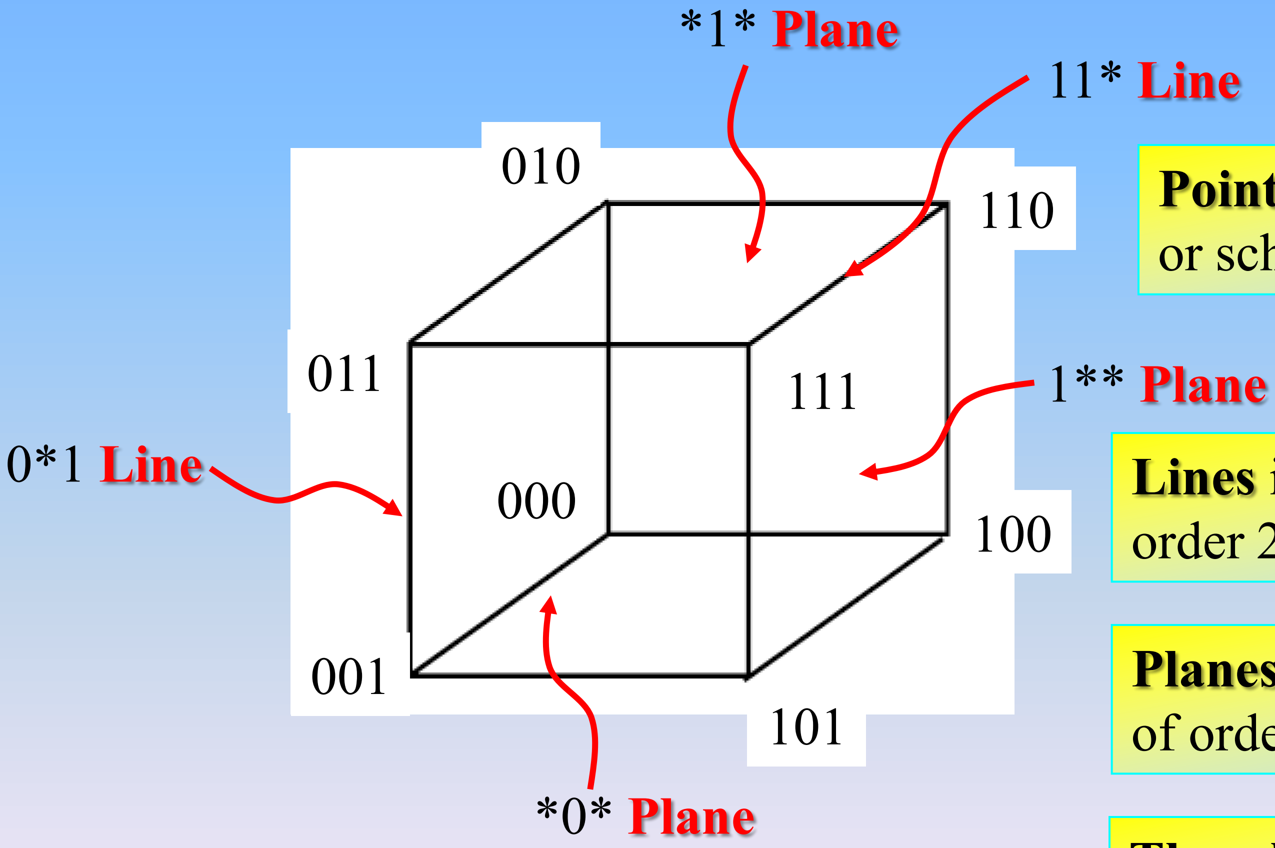


Points in the space are strings or schemata of order 3

Lines in the space are schemata or order 2

Visualization of Schemata as Hyperplanes in 3-D Space

Consider the strings and schemata of length $\ell = 3$



Points in the space are strings or schemata of order 3

Lines in the space are schemata or order 2

Planes in the space are schemata of order 1

The whole space is covered by the schema of order 0, ***

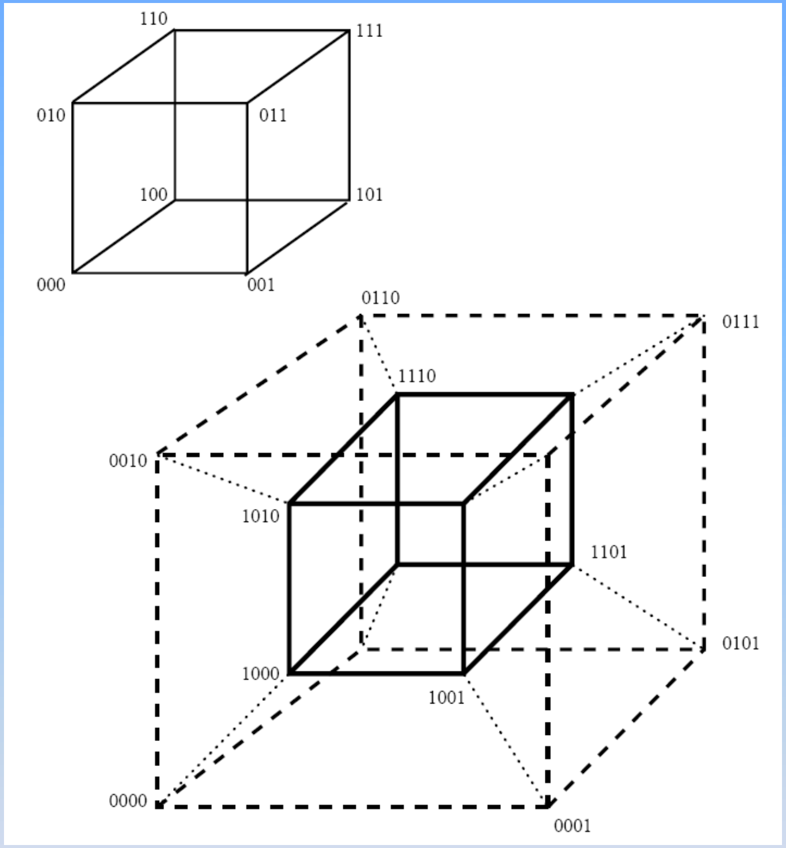
Similarity Templates as Hyperplanes

This result generalizes to spaces of higher dimension where we must abandon the geometrical notions available to us in 3-D space.

Points, lines, and planes described by schemata in 3-D generalize to **hyperplanes of varying dimension** in ***n*-space**.

We can think of a GA cutting across different hyperplanes to search for improved performance.

Schemata



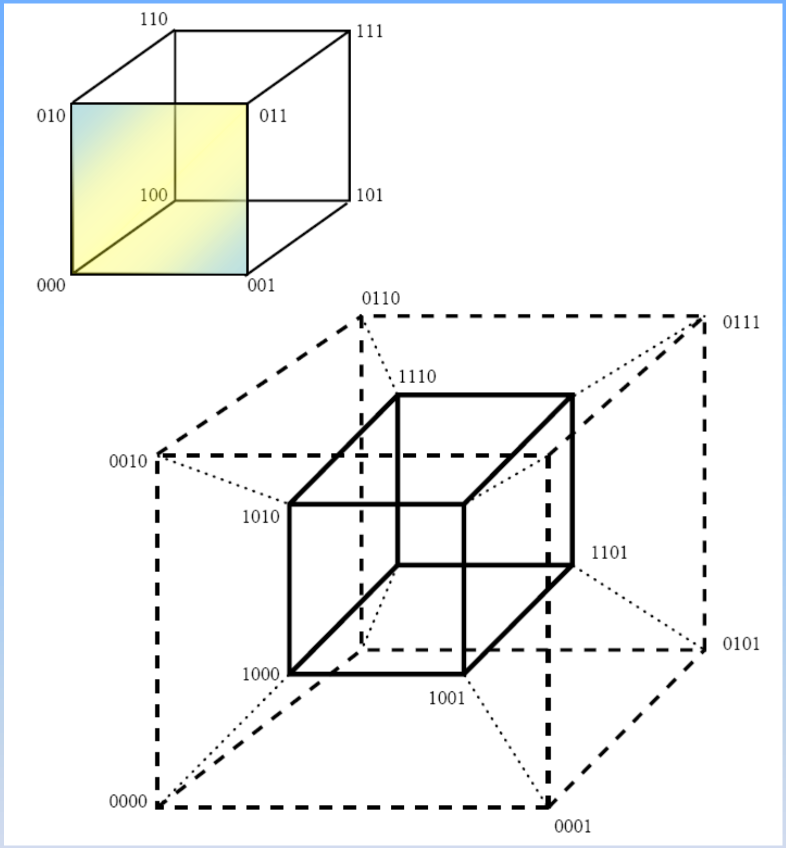
A **3-D cube** and **a 4-D hypercube**.

The corners of the **inner cube** and **outer cube** in the bottom 4D example are numbered in the same way as in the upper 3D cube except a **1** is added as a prefix to the labels of inner cube and a **0** is added as a prefix to the labels of the outer cube.

Only selected points are labeled in the 4D hypercube.

Assume we have a problem encoded with just **3 bits**, this can be represented as a simple cube with the string **000** at the origin.
The corners in this cube are numbered by bit strings and **all adjacent corners** are labeled by bit strings that differ by exactly **1bit**.

Schemata



- The front plane of the cube contains all the points that begin with **0**.

- If ***** is used as a **don't care or wild card match symbol** then this plane can also be represented by the special string **0****.

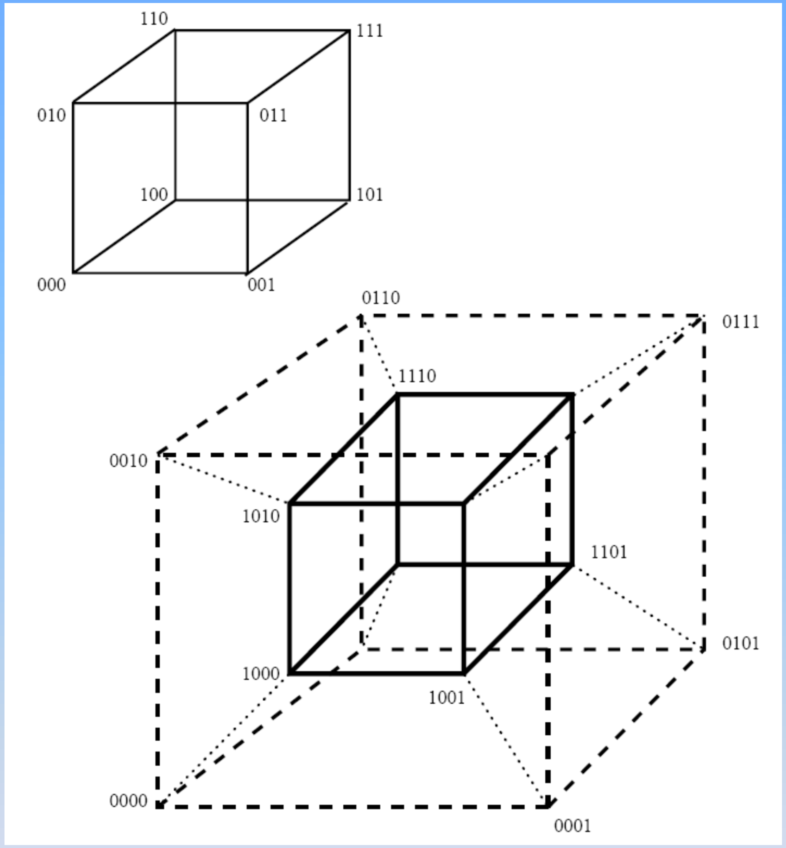
- Strings that contain ***** are referred to as **schemata**.

In general, for alphabets of cardinality **k,** there are $(k+1)^{\ell}$ schemata.

e.g. alphabets = {0,1} **k** =2
if ℓ =3 (3 bits),
then there are **27 schemata**.

Further, in a string population of ***n*** members, there are at most $n*2^{\ell}$ **schemata**.

Schemata

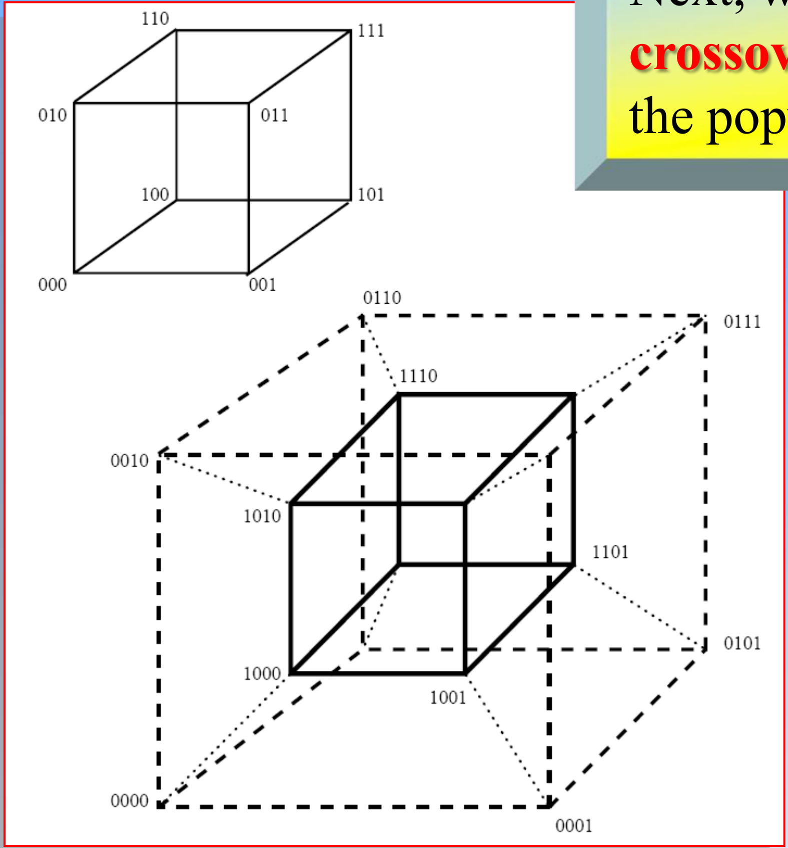


In general, all bit strings that match a particular schemata are contained in the hyperplane partition represented by that particular schemata.

This creates an assignment to the points in hyperspace that gives the proper adjacency in the space between strings that are 1 bit different.

- **inner cube:** corresponds to **1*****
- **outer cube** corresponds to **0*****.
- **fronts of both cubes:** ***0****
- **front of the inner cube**: order-2 hyperplane **10****.

Geometrical Vision of the Search Space



Next, we consider the operation of **reproduction**, **crossover**, and **mutation** on the **schemata** contained in the population

References

Genetic Algorithms: Darwin-in-a-box
Presentation by Prof. David E. Goldberg
Department of General Engineering
University of Illinois at Urbana-Champaign
deg@uiuc.edu

Neural Networks and Fuzzy Logic Algorithms
by Stephen Welstead

Soft Computing and Intelligent Systems Design
by Fakhreddine Karray and Clarence de Silva

References

A Genetic Algorithm Tutorial

Darrell Whitley

Computer Science Department