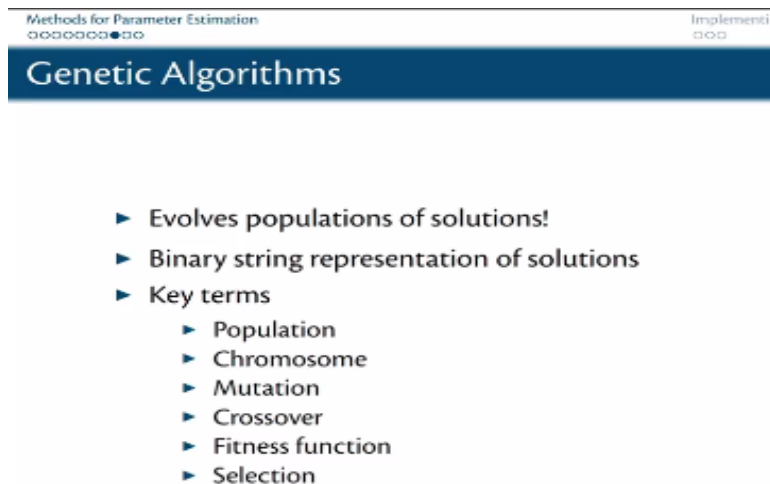


**Computational Systems Biology**  
**Karthik Raman**  
**Department of Biotechnology**  
**Indian Institute of Technology – Madras**

**Lecture – 47**  
**Other Evolutionary Algorithms**

So, in today's video, we will look at other evolutionary algorithms particularly we will look at a very interesting algorithm known as the differential evolution, so this is a very powerful tool for parameter estimation in biological systems especially and it is well implemented in this tool called PyGMO, which we will talk about in the next video and in today's video, I will also talk to you about implementing evolutionary algorithms, what are the aspects; various aspects that we need to consider.

**(Refer Slide Time: 00:48)**



Methods for Parameter Estimation  
○○○○○○●○○

Implementin  
○○○

## Genetic Algorithms

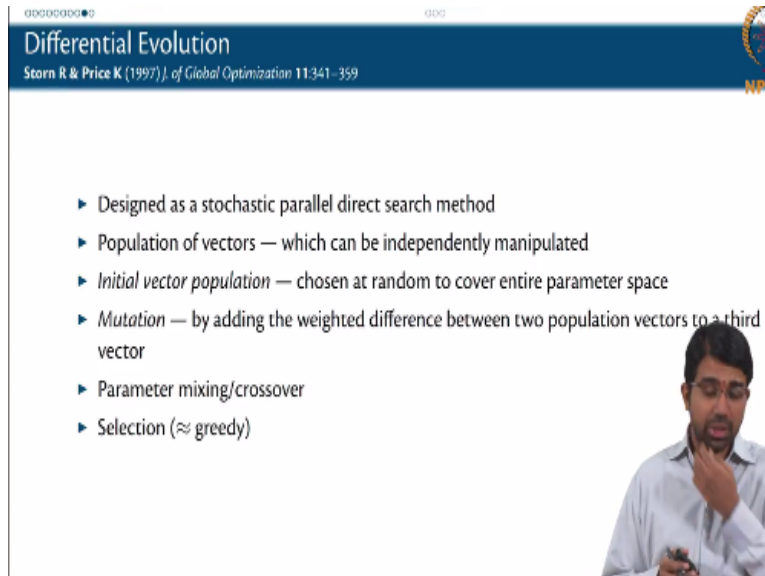
- ▶ Evolves populations of solutions!
- ▶ Binary string representation of solutions
- ▶ Key terms
  - ▶ Population
  - ▶ Chromosome
  - ▶ Mutation
  - ▶ Crossover
  - ▶ Fitness function
  - ▶ Selection

And we will also discuss as to when evolutionary algorithms are actually very useful as in when do you give up on the other kinds of algorithms and go in for evolutionary algorithms, okay, welcome back, let us continue with our study of some more evolutionary algorithms, to recap all the evolutionary algorithms, this slide is titled genetic algorithms but all evolutionary algorithms would fit this bill essentially, the evolve populations of solutions.

The representation will vary from one to the other and the key concepts remain the same, so you can talk about population, chromosome, mutation, recombination or crossover, fitness function,

selection and so on, so we already looked at some of these things, let us just look at a few more flavours of evolutionary algorithms.

**(Refer Slide Time: 01:30)**



The slide is titled "Differential Evolution" and includes the citation "Storn R & Price K (1997), of Global Optimization 11:341-359". It features a list of five bullet points describing the algorithm's characteristics and operations. In the bottom right corner, there is a small video inset showing a man in a white shirt looking thoughtful.

- ▶ Designed as a stochastic parallel direct search method
- ▶ Population of vectors — which can be independently manipulated
- ▶ *Initial vector population* — chosen at random to cover entire parameter space
- ▶ *Mutation* — by adding the weighted difference between two population vectors to a third vector
- ▶ Parameter mixing/crossover
- ▶ Selection ( $\approx$  greedy)

So, one important algorithm is known as differential evolution, this is designed as a stochastic parallel direct search method, so it is stochastic in that so, there is some randomness in how different points are picked and so on and it is of course a direct search method and it can be very nicely implemented in parallel, so thrives on a population of vectors, which can be independently manipulated.

So, which is similar to what most other evolutionary algorithms do or in case even genetic algorithms, right, so in genetic algorithms as well you have populations of bit vectors, here you have populations of potentially numerical vectors, you have initial vector population which is spread across the entire parameter space. So, you have a parameter space in  $r$  to the  $n$ , potentially, right, so you have vectors that cover different parts of the space.

And now you need to come up with ways to do mutation and cross over, right, you need certain random operators, you need a fitness function, you need a selection pressures or selection strategies and so on. So, how do you do the mutation, right? So, one way of doing a mutation is by adding some weighted difference between 2 vectors to a third vector, you pick 2 vectors at random, you compute the difference between those 2 vectors.

And add that difference to a third vector, right, so some multiplied by  $f$ ; some  $\alpha$ , right, so you weight that in some way, so add 5% of the difference, 20% of the difference to a third vector, you repeat this multiple times, you get more children; more population and then for crossover, what you do is; a little more simple, you do parameter mixing.

(Refer Slide Time: 03:28)

The diagram illustrates the Differential Evolution (DE) process. It shows two parent vectors, each with 8 elements labeled 1 through 8. A third vector is formed by recombining elements from the two parents. A box labeled 'Mutation' contains the formula:  $\vec{v} = \vec{v} + f(\vec{a} - \vec{b})$ . To the right, a list of search operators is provided: 'global' for mutation and 'local' for recombination/crossover. The NPTEL logo is in the top right corner.

Or essentially, you pick a random from each, right, so you mix the 2 to get a third vector, let us say this is 1, 2, 3, 4, 5, 6, 7, 8, I am just marking the positions, a, b, c, d, e, f, g, h, you might get a vector that is 1, b, c, 4, 5, 6, g, h, right, you randomly picked from; so you mix the parameters up, you can mix them in different ways, right, so this is the recombination operator in the case of differential evolution.

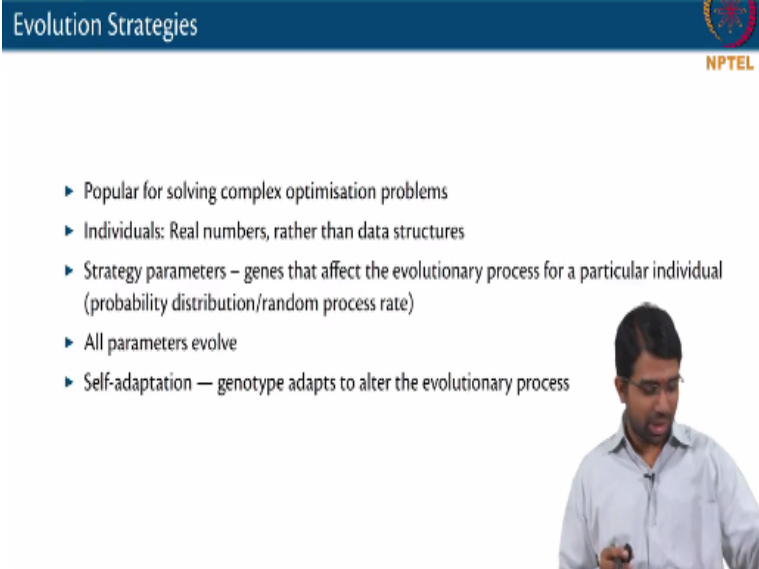
So, essentially you need the same thing, you need 2 or more if you can search operators, mutation, you can even think of this as global and local, right, 2 operators that look in 2 different styles on the search space that you have. Will it depend, right, so mutation; you cannot necessarily map into map or local but if you look at recombination, a sort of trying to take in; so in this case mutation will be more local in differential evolution.

Because you are like adding the difference between 2 vectors to a third and you know recombination could be more global in which because you are like picking a random pieces and

so on, were it really depends but neither of these would be purely global or purely local, right but together they do 2 different types of search essentially, very simple, so let us call this vector alpha, this vector beta.

So, you will say  $\gamma$  is  $\gamma + \text{some fraction} * \alpha - \beta$ , how do you do selection? You can again use several strategies, usual recommended selection procedure here is greedy, so you try to select more of the best solutions or pick the top k solutions and move on, right, this was the original differential evolution that was proposed by Storn and Price in 1997, there are many variances to it today.

**(Refer Slide Time: 06:10)**



The slide features a dark blue header with the text "Evolution Strategies" on the left and the NPTEL logo on the right. Below the header is a bulleted list of five points. To the right of the list is a small video inset showing a man in a light blue shirt speaking.

- ▶ Popular for solving complex optimisation problems
- ▶ Individuals: Real numbers, rather than data structures
- ▶ Strategy parameters – genes that affect the evolutionary process for a particular individual (probability distribution/random process rate)
- ▶ All parameters evolve
- ▶ Self-adaptation — genotype adapts to alter the evolutionary process

Some of them work very efficiently particularly, in case of biological problems. Evolution strategies are also popular for solving these kinds of optimisation problems in general, the major difference between evolution strategies and GA's or de; differential evolution is that real numbers are used instead of other data structures, right. See, if you see bit vector is actually a sort of a data structures in some sense, right.

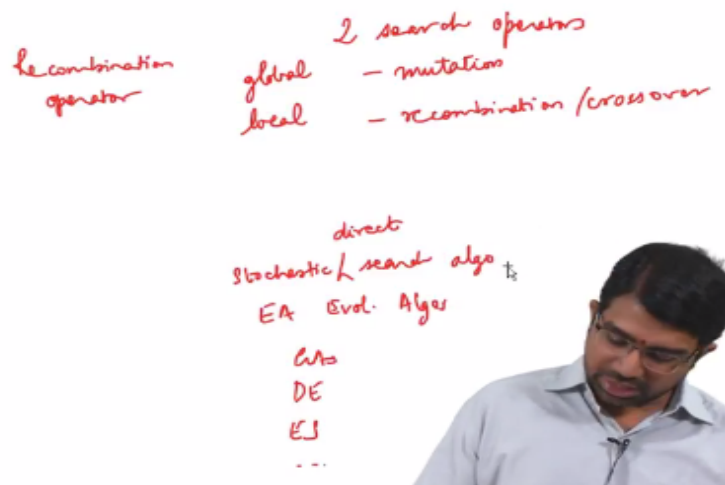
It represents something or it could even represent a circuit for all you know right, an electrical circuit as we discussed earlier and the interesting part is there are strategy parameters, these are the genes that affect the evolutionary process, so more biology in some sense, right, so these are genes, you can think of these as the genes that control the rate of evolution maybe the

recombines the you know, enzymes involved in SOS response or DNA replication and so on, right.

And even the parameters evolve, so your mutation rate and recombination the rate, they themselves evolve, so something like that and the genotype adapts to alter the evolutionary process itself, right, so the whole system keeps evolving at every point of time, right. So, essentially, if you will see the overall template is the same for what is; if you scale back a few steps, right, you said what are the different in the hierarchy of the algorithms.

**(Refer Slide Time: 07:38)**

NPTEL



We basically said GA's, DE's or DE evolution strategies whatever else, they all broadly come under evolutionary algorithms, these all broadly come under maybe direct search algorithms, if you see the template as the same for all direct search algorithms; be it stimulated annealing or any of the other methods are actually have a few more methods listed a little later on, we will come back to that. So, the recipe is the same and what is the recipe?

**(Refer Slide Time: 08:14)**

## Representation Paradigms

- ▶ Simple binary chromosomes
- ▶ Trees and complex data structures
- ▶ Cartesian Genetic Programming (for Evolvable Hardware)
- ▶ ...

You start with an initial population, maybe you know and you need to come up with the representation for the population that is the first catch, right, it is probably one of the hardest steps, right because this is where you need to think beyond that it is all just the recipe, the computationally, what is the hardest step; computing the fitness function over and over again because you want to calculate the fitness of the entire population.

So, you need to compute the fitness function over and over again, so that is a problem, right, so you can have trees and or other kinds of complex data structures, you can have Cartesian genetic programming that something I mentioned earlier so on and so forth, so you may have, for scheduling you may have a very different kind of data structure, right, so you need to come up with representation paradigms.

**(Refer Slide Time: 09:07)**

- ▶ Macro-mutation: Large change in alleles without recombination
- ▶ Hybrid operators (not typical of evolution at all), e.g. Hill climbing
- ▶ Operators for permutations (e.g. Travelling Salesman)
- ▶ Learning — individuals alter their chromosomes before replication
- ▶ Evolving operators (e.g. by encoding probabilities into the chromosomal representation)

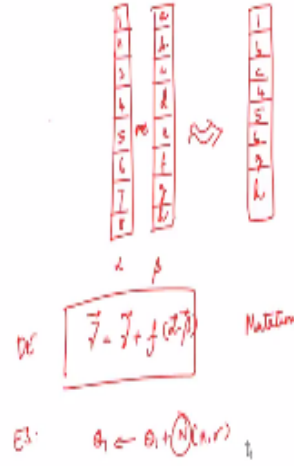


And you need search operators; you can have macro mutations, so large changes in alleles without recombination, so mutate several genes, several bits, right. Hybrid operators that you know do not really have anything to do with evolution, you can have like a hill climbing operator that will just you know increase the value in a particular direction, right, it does not have to truly honour the evolutionary process.

We can have operations for permutations, very useful in case of travelling sales person and problems such as those. Learning operators, where alter their chromosomes before application based on some parameters and so on and evolving operators by encoding something like mutation probability into the chromosomal representation itself, right, so this is where we talked about self-adapting chromosomes and so on.

**(Refer Slide Time: 10:09)**

DE



Recombination operator

2 search operators

global - mutation

local - recombination / crossover

don't

Stochastic search algo

EA Evol. Alg

GA

DE

ES

..



So, essentially you need operators right and there is a big library of possible operators to choose from, another very simple operator is; so, this is a simple operator for, this is for the evolution strategies very simple operator is let us say there is a parameter theta 1, just add a randomly normally distributed variable to it, it is already a search operator, so this is actually a very simple search operator and evolution strategies use such simplistic search operators.

But you can draw it from any interesting distribution not necessarily just normal, so you can have different kinds of operators as well, right and yeah, you had a question, **“Professor – student conversation starts”** so, what is the difference between learning and mutation, when can you learn, you need a teacher, right, so you basically need something that tells you whether you are doing well or not, you need some evaluation cycle, right.

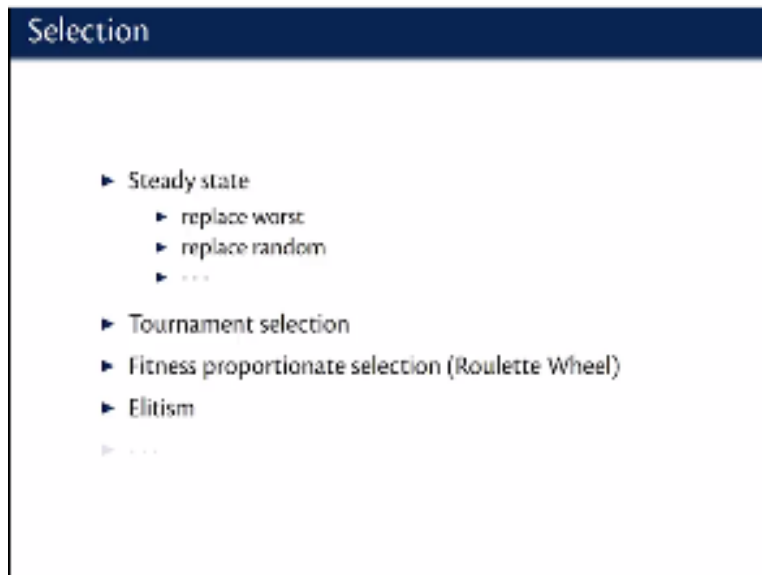
So, like a classifier you need to basically let us say, you build a classifier, when do you say it has learned, you basically test against some data and see that the performance has improved, after you have added some more data to the training kind of thing, whereas here this evolving operators could just be arbitrary, let us say I have a chromosome that encodes along with all my thetas and so on, it also encodes a hyper parameters, then like randomly just around, right.

I am going to randomly mutated, so maybe I mutated the hyper parameter instead of mutating my parameter, the parameter we are trying to estimate here, so that is a possibility, whereas in a



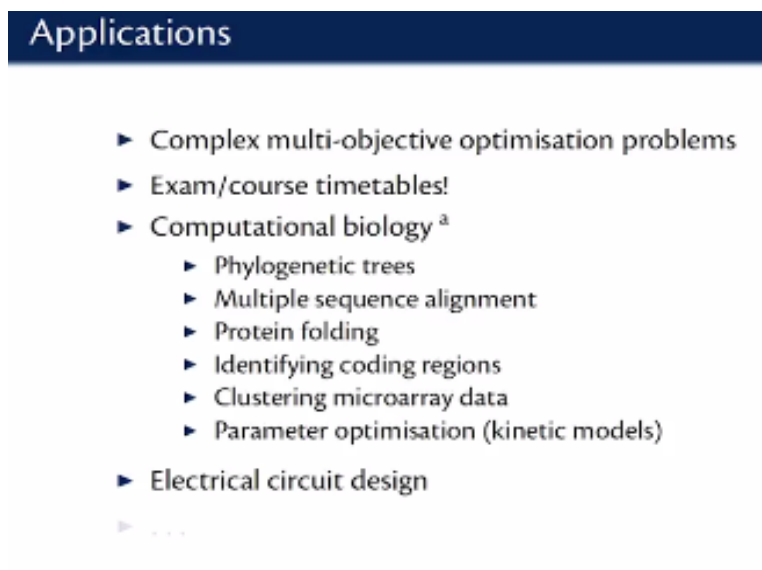
sense there is no direction, whereas learning will try to proceed the direction of improving something. **“Professor – student conversation ends.”**

**(Refer Slide Time: 12:12)**



And selection, so you can try to you know replace worst or replace at random, you can have tournament selection, you can have fitness proportionate selection, you can have elitism and anything else that you can actually come up with, you can use different kinds of selection, pressures or selection methods.

**(Refer Slide Time: 12:37)**



So, to summarise the main applications of GA's or in; or evolutionary algorithms or in complex multi objective optimisation problems, right so to be exam course timetables or any scheduling

problem as we saw or in computational biology, it is used in phylogenetic trees, multiple sequence alignment, protein folding, identifying coding regions and proteins, clustering microarray data, for k- means clustering you can imagine that this would be very useful.

Because it is k- means clustering is again an optimisation problem, parameter optimisation for kinetic models which is where we started looking at it, it could also be use for things like electrical circuit design and so on.

**(Refer Slide Time: 13:17)**

**Conclusions**

**When to use Evolutionary Computation?**

- ▶ Often quite useful
- ▶ Careful choice of representation, operators etc. is crucial
- ▶ Especially useful, when structure of search space is poorly understood
- ▶ *Might help understand the problem better*
- ▶ No reason to believe that a GA approach would be any better than any other optimisation technique!

**Challenges**

- ▶ Choice of representation
- ▶ ~ Black box behaviour
- ▶ Computationally expensive, esp. fitness calculations

Okay, so when to use evolutionary computation? They are often quite useful, what are their strengths or what are the challenges; one major challenge or one thing that you have to usually worry about is; you need to have a careful choice of representation operators and so on but is very useful especially, when you do not understand the space, you have really a weird space, you do not know what is going on in that space.

You do not know what is the gradient, you do not know what is the landscape looking like, GA's are useful, it might help understand the problem better, once you solve a problem using GA's, you may be able to extract some principles out of it that is a possibility but there is no reason to really believe that GA approach is better than any other organisation technique, so and often times, although sometimes it might help you understand the problem better.

A lot of times, you do not understand what is going on, it is literally like a black box, you throw in some parameters something else comes out, they are good, you are happy but you do not know you got there, so the choice of representation remains the biggest challenge and of course, I already mention the black box issue and it is of course very expensive because you are evaluating 100's of 1000's of solutions, fitness function evaluations, right.

**(Refer Slide Time: 14:51)**

### Recent advances/Related techniques

- ▶ Gene expression programming
- ▶ Simulated Annealing
- ▶ Ant Colony Optimisation
- ▶ Particle Swarm Optimisation
- ▶ Tabu Search
- ▶ Interactive EAs

A population size might be 100 and it might run for 10,000 generations and so you are looking at a huge number of millions of fitness function calculations. Other related techniques again you know similar recipes will be followed, gene expression programming, stimulated annealing, ant colony optimisation, particle swarm optimisation, Tabu search, interactive evolution algorithms and so on.

There are various flavours to evolution algorithms and all of these are useful for many kinds of complex optimisation problems particularly, parameter estimation. So, if you look at parameter estimation invariably in any paper you would find that they would have gone beyond the standard (()) (15:25) Matlab reference search because it is just not good enough to work in large spaces.

**(Refer Slide Time: 15:39)**

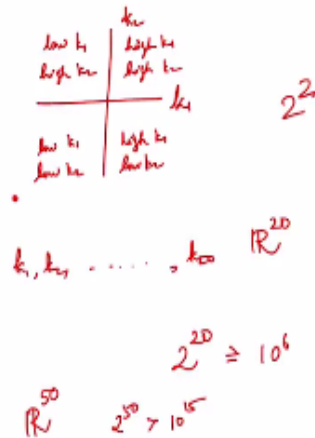
## Curse of Dimensionality

$$|O| \approx 20^{15}$$

$$30 < k_1 < 30000$$

$$0.1 < k_2 < 10^2$$

$$k_3 \approx 520 \pm 10$$



Because the most important issue we have to worry about in large space is actually a curse, right, it is called a curse of dimensionality, what is the curse of dimensionality? It is basically telling you that there is an inherent problem in very high dimensional spaces, so it is a simple example, let just say this is some  $k_1$  and  $k_2$ , you do not take this as a origin may be origin let us say somewhere here, right.

So, I would say this is low  $k_1$ , low  $k_2$  and this would be high  $k_1$ , high  $k_2$  and this would be low  $k_2$ , high  $k_2$  and this would be low  $k_1$  high  $k_2$ , to examine 4 combinations; to study 2 parameter problem, you need about at least 4 regions to explore that is 2 to the 2, suppose you have  $k_1, k_2, k_{20}$  or searching in  $\mathbb{R}^{20}$  and you need 2 to the 20 regions to explore, you want at least one representative from each region to understand what is happening to my system.

And this is already 1 million +, right and what happens if; we are talking about 50 parameters, right, it will be, so 2 to the 50 will be  $> 10$  to the 15, so if you go to the larger and larger problem in the sense, you will quickly reach the number of atoms in the universe, right, so basically these are very difficult problems to solve and it is an inherent problem, it is not that if you build much better supercomputers tomorrow, you will be able to solve this problem, right.

It is a fundamental problem in such spaces, where you have very large number of parameters such to explore or very large spaces to explore, right so this is something that you need to really

worry about. So, in practice the size of your parameter set, maximum some 20, 25, if you are doing an unconstrained search but usually you start restricting the parameters, so I know that you know,  $30 < k_1 < 3000$ , this is already a great thing.

Because you are restricted to 2 orders of magnitude only, 3 orders of magnitude, right or you know that  $0.1 < k_2 < \text{you know } 10 \text{ to the } 2$ , a few orders of magnitude and this is already going to be very helpful, so typically all parameter estimations have to be very constrained otherwise, there is no way you are going to get it, best thing is if I know that  $k_3 \text{ approximately } = 520$ , let us say great.

**(Refer Slide Time: 19:26)**

### References/Further reading

- ▶ **Foster JA** (2001) *Nat Rev Genet* **2**:428–436
- ▶ **Storn R & Price K** (1997) *J. of Global Optimization* **11**:341–359
- ▶ **Ab Wahab MNN et al.** (2015) *PLoS ONE* **10**:e0122827

So, it means that may be it is only + or -10 or something like that right, so if the tighter you know certain parameters the better it is, so these are some very good paper that you should consider looking at especially, the first one. It is a; it is some nature review genetics finally and it talks about also evolutionary algorithms, so it is a very long, it is a very nice, you know detailed write up about several evolutionary algorithms.

**(Refer Slide Time: 19:55)**

## Recap

### Topics covered

- ▶ Differential Evolution
- ▶ Implementing EAs
- ▶ When to use EAs?

### In the next video ...

- ▶ Introduction to PyGMO
- ▶ PyGMO Algorithms and Examples
- ▶ Multi-objective Optimisation

It is actually a shortest paper only 8, 9 pages and but it lists every flavour of evolutionary algorithms. In today's video, I introduced you to this very interesting evolutionary algorithm known as differential evolution, there are many variants for DE as well and I also introduced to you to how we go about implementing evolutionary algorithms, now what are the different kinds of selection, methods or other kinds of search operators and so on.

And we also discuss as to when one actually goes in for evolutionary algorithms and then you know the other kinds of algorithms are no longer useful and one has to resort to stochastic direct search algorithms particularly, like evolutionary algorithms. In the next video, I will introduce you to this very interesting tool called PyGMO, which is a Python based library for doing multi objective optimisation and so on.

It supports various kinds of genetic algorithms and so on, so I will talk to you about what are all the different kinds of algorithms that supports and some examples are you know how to we actually use PyGMO for solving or estimating parameters and so on and also give you a brief introduction to multi objective optimisation, what happens if you have more than one objective to optimise.