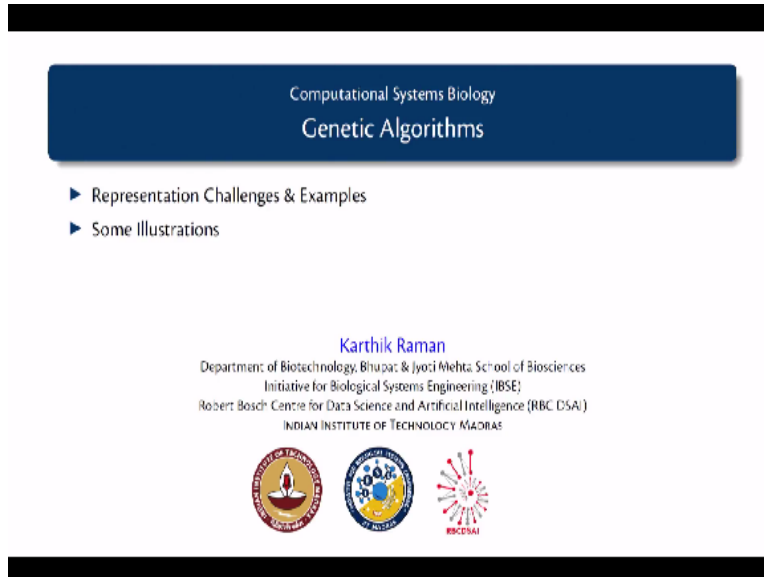


Computational Systems Biology
Karthik Raman
Department of Biotechnology
Indian Institute of Technology - Madras

Lecture – 46
Genetic Algorithms


(Refer Slide Time: 00:11)



Computational Systems Biology
Genetic Algorithms

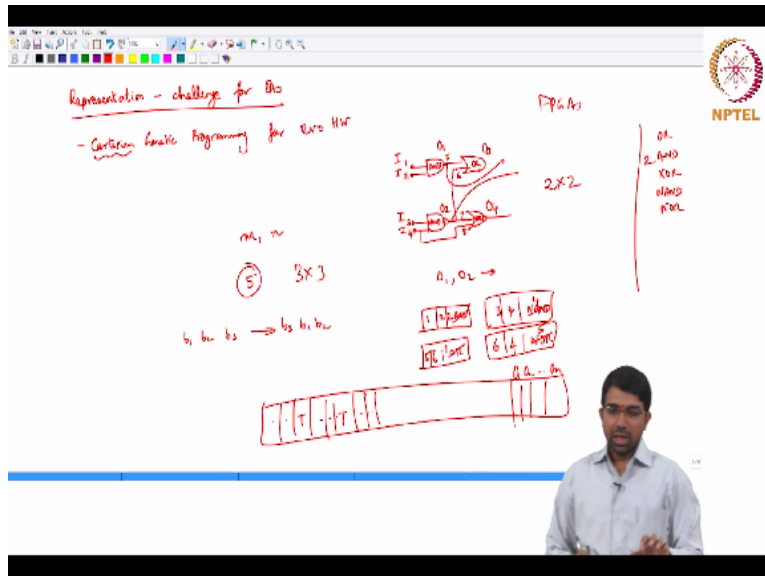
- ▶ Representation Challenges & Examples
- ▶ Some Illustrations

Karthik Raman
Department of Biotechnology, Bhupat & Jyoti Mehta School of Biosciences
Initiative for Biological Systems Engineering (IBSE)
Robert Bosch Centre for Data Science and Artificial Intelligence (RBC DSAI)
INDIAN INSTITUTE OF TECHNOLOGY MADRAS



In this video, we will continue with our discussion on genetic algorithms and importantly, we will look at what are the challenges in representing data so that it is readily amenable for a genetic algorithm and we will also look at an illustration or a simple example to see how one can potentially use GAs and how they can, you know, lead to convergence to the desired minimum.

(Refer Slide Time: 00:33)



Again, let me call this EAs, right. So this is a very interesting representation for a relatively complex system which is called, so the representation is Cartesian. It is called genetic programming for evolvable hardware. So let us say you have a system like this. It is a 2*2 array of gates, right. So I am just giving this, this is not you know, a very general example to what we are studying here.

But it will give you an idea of how do you construct, creatively construct representations for a difficult search problem. This is what you need to do if you are working with genetic algorithms, right. So what do you do? So what gate is this? AND or NOR, NAND, right. So let us call these, let us have a list of gates or AND, XOR, NAND, NOR. Any other gates commonly used. You have XNOR I guess. NOT, yes.

I am trying to restrict myself to 2 input gates, right. This is input 2, this is input 3, this is input 4. This is, I will call this 5, 6, 7, 8 and O1 and O2, something of this sort, right. So now I need a representation. So I can go and evolve, let us say I want, should follow a particular truth table. So I am going to find a function that does, I am going to find a gate arrangement that does something, right.

So this is actually how people evolve certain very complex circuits. Here is a very interesting, this was done by Adrian Thompson back in 1995, right and what they found was evolution,

evolutionary algorithms were taking advantage of things that normal designers do not do because it is, it is a more blind thing, right. So it essentially tries to tap into a very different portion of design space even, right because the way, so in fact somebody else was mentioning to me recently, they should go and look up the NASA evolutionary design antenna.

It apparently looks very ugly but it is very effective. It has got a very weird shape because it was evolutionary design person designed by a designer but the antenna is still very very effective because well it was optimized very carefully. So how do you represent this? I can represent every gate by, let us start with this. So I would say 1, 2 and then 3, 4 NAND, then 5, 6 OR, then 7, 8 NOR.

I could stick all these into a row, that is what we call a Cartesian representation and I will say AND I will replace it by 2, OR I will replace it by 1, NAND I will replace it by 4, NOR I will replace it by 5. So a representation for this gate would be, and my outputs are, so I will call this O3, O4, O1 and O2. In fact, I do not even, yes, I need to just come up with some, right, so I could call this 5 and 6 or whatever.

The what output goes into the next one, right. So I need to give this a number 2. So these are the inputs that are going in here. This same input is also going here. So I would call, so this actually will not be 7 8, this will be 7 4 and not 7 4, it will be 6 4. Is that right? So this is, I call these, so just think up. You just need to come up with your, with, you know, a systematic rule for representing, right.

If you have an M cross N array, right. If I say 5, I know where it is. So if it is a 3 cross 3 array, I know it will be in the second row second column, that will be 5, right. If it is a 6 cross 6 array, I know it will be in the first row fourth column, something like that. So I just need a long representation which now says what is the input to each gate and what is the type of the gate.

Input input type of the gate, input input type of the gate and finally, what are the outputs. Now all I have to do is, I can just go and mercilessly change things here. I will get new gate configurations. I can simulate them and I can see which direct, am I getting a better function or

not. For that, what will I have to do? I have to do fitness and all that stuff.

(Refer Slide Time: 06:48)

The slide contains a diagram of a chromosome with genes labeled $B_1, B_2, B_3, \dots, B_n$. A red arrow points from the top chromosome to a bottom chromosome, indicating a crossover event. Handwritten notes include "X Crossover" and "✓ Direct mode". To the right, a list of evolutionary concepts is written: "Chromosome (Individuals)", "Population", "Mutation", "Crossover", "Selection", and "Fitness (Function)". The NPTEL logo is visible in the top right corner. A person is partially visible at the bottom right of the slide.

So we will have to go back to all of these concepts.

“Professor - student conversation starts” (()) (06:50) you could, you could, you could. I will now take output from the second gate itself, may be, I do not know. I could take any of this and connect it to output. I may, I may just ignore this. (()) (07:09) if it does not matter (()) (07:11). Yes, you just want (()) (07:13) to follow whatever true table you have set. I might want several shift functions, meaning if I give $B_1 B_2 B_3$ as input, I need to get $B_3 B_1 B_2$ as output, that is all.

Any bunch of gates, any output, whatever does this job, I am happy with it. This is actually very interesting if you go and look this up because if I, we did a study back in 2011 where we showed that we can, this is something we discussed in the very end of the course. Then we talk about robustness and evolvability. So we showed that we could actually design robust electrical circuits as well.

So these are typically can be easily implemented on FPGAs. FPGAs are field programmable gate arrays meaning they are not setting, you know, setting stone as an NAND gate or OR gate or something like that. Using a configuration string, they can be dynamically reconfigured as a NOR gate and NAND gate and so on. “Professor - student conversation ends.”

So you take this particular string potentially, you have to do lot more work than that and you feed it to the FPGA. It will now take up assume this circuit. You change the string, it will assume a different circuit. So it can reconfigure itself. So can you design a circuit that can easily from making a just a small rewiring, convert itself from computing a circular shift to a right shift or you know, you can even think of more harder functions, right.

So if you really want to extend, extend, extend this, can you build a reconfigurable robot because it is a very popular application. There are some reconfigurable robots that people have built for say deep sea exploration, right. There are some kind of (()) (08:50) has to seek under titanic, it will just, you know, become, it will crawl and go. It will be walking as a (()) (08:56) but then it has some real obstacle in the way, it will reconfigure, become a reptile and crawl through, something like that.

So there is some example of this sort but basically how do you design such circuits? Because the search space is so large, you have to really resort to some very intelligent algorithms and evolution seems to work very well with the very large search space, right. That is, you know, sort of the observation that we have from observing years of biology, right. Evolution seems to work well in practice, right.

So why does it work? There is a selection pressure, right and there is some notion of fitness and then you basically, you actually do not worry about how evolution happens in real life, right. You might have heard on; see we basically mostly agree with the Darwinian theory of evolution. There is also the Lamarckian theory of evolution. As far as GAs are concerned, they are as happy with the Lamarckian theory as with the Darwinian theory as long as it can give you better results, right.

That is all you really worry about. So if you look at this framework that we are talking about, how do we go about and compute each of these things. So let us take another simple example. So now I think you have some idea of how would you do for scheduling? You just need to establish these operators and how would you do it for, you know, a bunch of electrical circuits.

(Refer Slide Time: 10:22)

The slide shows handwritten notes on a whiteboard. On the left, the equation $x^2 = 9$ is written. Below it, a list of four 3-bit strings: 000, 001, 100, 111. An arrow points to a box containing these strings, with the text "pop-size = 4" above it. To the right of the box, another list of strings is shown: 001, 101, 110, 101, 110, 000, 001, 100, 111. Each string is associated with a fitness value: $\frac{1}{9}$, $\frac{1}{9}$, $\frac{1}{9}$, $\frac{1}{9}$, $\frac{1}{9}$, $\frac{1}{9}$, $\frac{1}{9}$, $\frac{1}{9}$, $\frac{1}{9}$. A box highlights the string "100" with a fitness of $\frac{1}{9}$. Below this, the text "Fitness: 100, 100, 100, 100" is written. At the bottom, the text "Evolution: Fitness proportional" is written. In the top right corner, the NPTEL logo is visible.

If we look at a very simple example, let us say I want to solve $x^2=9$. You already know the solution. Let us assume that the solution is only in integers, right. So I will have to now assemble a bunch of candidate solutions. So $x^2=9$, let me say I will use 4 bits to represent the solution, may be 3 are, or I will use 3 bits to represent the solution, right. So my possible solutions are these.

Many more, right. Let us just start with some subset of this. So I start my population size, I take it as 4, right and the numbers I use are 000 001 100 111. We have carefully left out the solution from this but let us say this is our initial population. Now what do you do? No before that. You evolve. You, you mutate, right. So let us say I mutate this bit, right. So it becomes 001, 1 and let us say here I mutate this bit, this will become 1, wait, I do not want to get to the solution.

Let us mutate this bit, so it becomes 101. No, no, no, you do not care. It is random. It is blind. That is very important, right. So that comes back to the question that you were asking yesterday, what is parallelisability. Just parallelly do this. Throw it to 100 course, just change it. You do not care what is happening. All you care is, I will now generate some large number of individuals and I can squash it to the original population size.

Finally, I have to somehow come back to $N=4$, right. I do whatever, right. So this 100 might

probably become 110 and 110 may become 011, right and let us do a few crossovers. Let us crossover 001 and 100 and we might get a 101 again from that and let us cross 111 with 000, you will get 110 let us say. You did a very small example, so you are not going to get in a variety. First of all, your space has only 8 numbers, right but given this, so now we have 6 individuals, squash it, in fact, you should, there is a very systematic way to do this.

You might actually end up with exactly 8 individuals or something like that. So you will end up with 8 individuals, you do 2 mutations and 2 crossovers or something like that and including the parents, you will have 8 individuals. Now you compute the fitness for each of these. What would the fitness here be?

“Professor - student conversation starts” (()) (13:25) So may be something like, but closer to 0, higher the fitness. Closer to... because your best solution is $x=3$. So 3^2 has the highest fitness. So you may want to do $1/x$ or something or you can do - whatever. Let us just say $1/x$, right. **“Professor - student conversation ends.”**

So $1/x^2$ is my fitness value. So this is 1, so it will be $1/8$ and I can take absolute value also, it does not really matter. So this is, what is this? This is phi, you know. So $1/16$. This is, what is this? 6. So this is $1/27$. This is 3, okay. You have infinity, I do not want this. Let us leave this out, right because we want to do more iterations and this is again $1/16$. This is again $1/27$. How do you pick the best solution now? How do you pick the new population now?

“Professor - student conversation starts” Parent also, you will have to evaluate. Yes, you have to evaluate for the parent also. So this will be $1/9$ for 000. 001 will be $1/8$. 100 will $1/5$ and 111 will be $1/40$, right. So now I have all the fitnesses. Tell me one to pick. 100 (()) (14:59). 100 will be $1/5$, 100 is 4, no, so 7, yes. Sir is the mutation possible with (()) (15:07) Why cannot we just eliminate the (()) (15:17) My absolute solution.

That does not mean from... No why, why are you throwing away any information. So may be hold on to your information. How do you do selection? How do you use the fitness to make a selection? (()) (15:34) so one way you can do is, you go for what is known as an (()) (15:40)

selection, meaning what is the best guy here? $1/7$ and you just make copies of this. This is one way.

This is called (()) (15:57) selection. What is the problem with this? Is there a problem with this? (()) (16:00) This is like as aggressive as hill climbing. You know, I will only take the best and like I am forgetting any other variety that I have and so on and so forth, right. So that is a potential problem with this, right. **“Professor - student conversation ends.”**

On the other hand, you can do something called tournament selection. Meaning you pick a bunch of solutions. What do I mean by bunch? It is a hyperparameter. You have to choose that bunch. So what is the size of that bunch? That size could be 2, that size could be 1, that size could be 5. So you pick a bunch of solutions and then you, you basically have them play a tournament against each other, like a fitness tournament and you pick 1 of them or 2 of them.

So you can choose something like that, right. That is one way. The best way is something known as roulette or more technically may be Barabasi-Albert type, right. So you pick a solution in probability proportional to its fitness. Higher the fitness, higher the probability. So you could probably just normalize all of these. Think of these as your degrees for your Barabasi and you would compute the probabilities, right.

So, so $1/8/\sigma$ fitness will give you fitness of this. So you pick that with a particular probability. So better solutions, this is a very Darwinistic, right. So higher, better individuals with higher fitness have higher probability of survival, survival of the fittest. So all of these will have advantages and disadvantages. And I can give a name to each of those things, you know, somebody would have already come up with this strategy.

So there is no, I think, I want you to particularly embrace the concept here rather than the exact algorithm. Because I do not think there is a genetic algorithm. There are genetic algorithms, right. You can mix and match any of these recipes to come up with something that is best suited to your problem. What kind of fitness, I mean, sorry, fitness is the one thing that is non-negotiable, right?

Fitness is well defined by your, so here in our case $e^{-\theta}$ will be the fitness or $-e^{-\theta}$ whatever, right. That will be your fitness. Where ever, but in this, in any, in every different case for IPL, there will be a different kind of fitness. For some other TSP, there will be a different kind of fitness, so on and so forth. So the important concept to remember is that you need these operators that cause variation. So how does this compare with what we asked off yesterday.

(Refer Slide Time: 18:42)

The slide is titled "Methods for Parameter Estimation" and includes the citation "Desirable Characteristics Storn R & Price K (1997) J. of Global Optimisation 11:341-359". It features the NPTEL logo. A list of characteristics is shown:

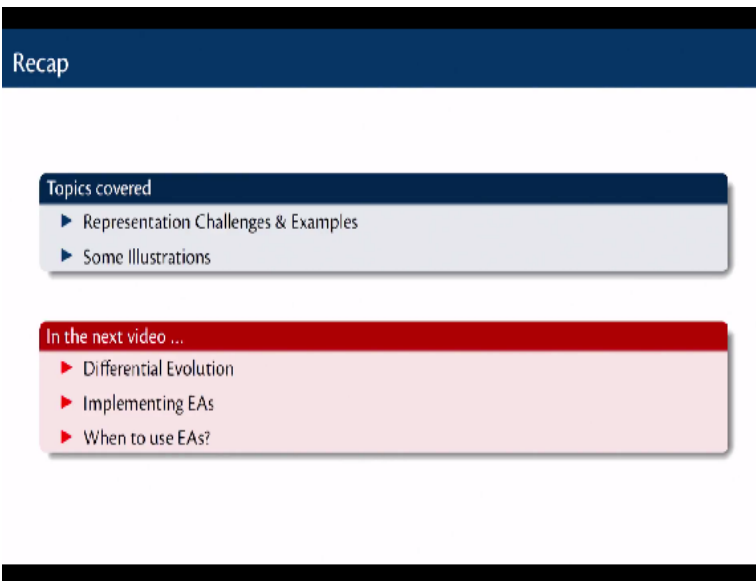
- ▶ Ability to handle non-differentiable, non-linear and multi-modal cost functions
- ▶ Parallelisability to cope with computation intensive cost functions
- ▶ Ease of use, i.e. few control variables to steer the minimization
- ▶ These variables should also be robust and easy to choose
- ▶ Good convergence properties, i.e. consistent convergence to the global minimum in consecutive independent trials

A video inset shows a professor in a light blue shirt gesturing with his hand.

Ability to handle non-differentiable, non-linear multi-modal cost functions, parallelisability, few control variables. What are your control variables? **“Professor - student conversation starts”** (()) (18:52) number of iterations. That is number of, number of generations, population size. Population size. Method of selection. Okay, even before that. Mutation rate crossover. How often do you mutate? how often do you crossover.

Sir but mutation after every generation (()) (19:16). Yes, yes but how many individuals you will mutate, right. So you typically like in biology you say, you say $N_{\text{new}}=10$. What is $N_{\text{new}}=10$ mean? 1 mutation is 10 individuals. Right. So for a population of size N , I will mutate 10 people. So I will mutate 10 people every generation. $N_{\text{new}}=10$ means. $N_{\text{new}}=1$ is actually very, very special case. $N_{\text{new}}=1$ has very different evolution characteristic, evolution trajectories compared to other things. **“Professor - student conversation ends.”**

(Refer Slide Time: 19:49)



Recap

Topics covered

- ▶ Representation Challenges & Examples
- ▶ Some Illustrations

In the next video ...

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

In today's video, I hope you got an introduction to the challenges that are involved with representation and some example representations and also an illustration of how we go about, you know, solving the genetic algorithm. In the next video, I will introduce you to another evolutionary algorithm namely differential evolution which is a very powerful algorithm, especially for handling biological datasets of our biological parameter estimation and so on and we will also look at some of the challenges in implementing evolutionary algorithms and we will discuss as to when EAs are actually useful.