

Artificial Intelligence: Search Methods for Problem Solving
Prof. Deepak Khemani
Department of Computer Science & Engineering
Indian Institute of Technology, Madras

Chapter – 04
A First Course in Artificial Intelligence
Lecture - 30
Population Based Methods:
Genetic Algorithms and SAT

(Refer Slide Time: 00:14)



Genetic Algorithms

Survival of the Fittest

Competition

Evolution



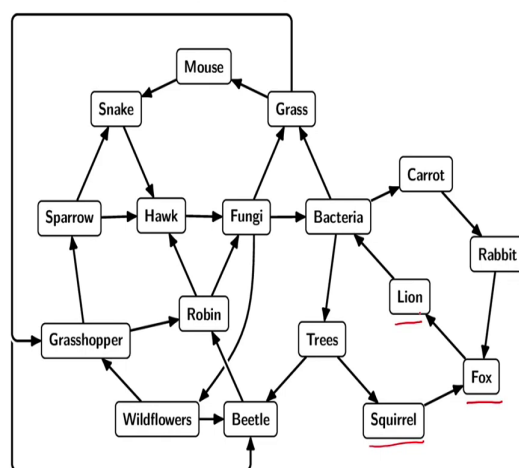
Welcome back. Let us continue our study of Genetic Algorithms which you may recall is optimization method inspired by nature. Because nature kind of keeps improving the design of life forms of creatures and it does it through a process of natural selection and the genetic algorithms are kind of inspired by this process.

So, if you recall, we said that the key feature in evolution is survival of the fittest and there is competition for resources, scarce resources amongst life forms which we call as the phenotypes and this competition leads to selection.

And this selection of the fittest or survival of the fittest leads to a continual improvement in the life forms, the ability to survive essentially ok. So, there is the practically a arm race between the predators and the prey, the predators become more and more adept at catching their prey and the prey at the same time, learns ways to escape from the predator and somehow, they all exist in some kind of a Cauchy's stationary equilibrium.

(Refer Slide Time: 01:42)

An Ecosystem of Predators and Prey



A small fragment of the vast ecosystem. The natural world contains millions of species interacting with each other. Arrows depict a positive influence of the population of one species on another.



So, we saw this diagram in the last video. This is a slightly monitor version of the same diagram. Thanks to Baskaran and what this depicts is a small fragment of the ecosystem,

where the different species which are inter dependent survive together and the whole ecosystem kind of thrives essentially.

So, the arrows here depict influence on population. So, such a diagram is often called as an influence diagram and we can see that for example, the population of squirrels is directly influences the population of foxes, assuming that foxes eats squirrels and the population of foxes in turn is influences the population of lions.

So, the influence here that we have depicted is positive influence and there would be negative influences also which could be roughly in the opposite direction of the arrows that we have drawn. So, there is this complete ecosystem in which all these different species, they are fighting for survival and all of them exist together and the whole world kind of goes on.

(Refer Slide Time: 03:07)

Evolution

Species evolve continually.

A species is like the design of a life form.

Nature has evolved a process of creative adaptation

In his famous book "*The Blind Watchmaker*" the British evolutionary biologist Richard Dawkins has emphatically argued that the "*ratchet mechanism*" in nature (remember – *Things that persist, persist...*) has been responsible for the evolution of complex life forms, rather than a supernatural creator....



So, evolution is a process by which all these species, they evolve continuous continually. A species is like we said a design of a life form and nature has evolved a process of creative adaptation to keep improving upon this design. Now, there used to be alternate theories of the world or philosophy you might say and one of the competing theories was that of creation or creationism which says that god created the world and that is why it is so perfect essentially ok.

So, much so that when Charles Darwin came out with his theory of evolution, he was somewhat reticent to publish it because he was worried about the response that he would get from society at large and the church in particular. But eventually, he did published his work and now it is widely accepted theory of how different life forms came to be essentially.

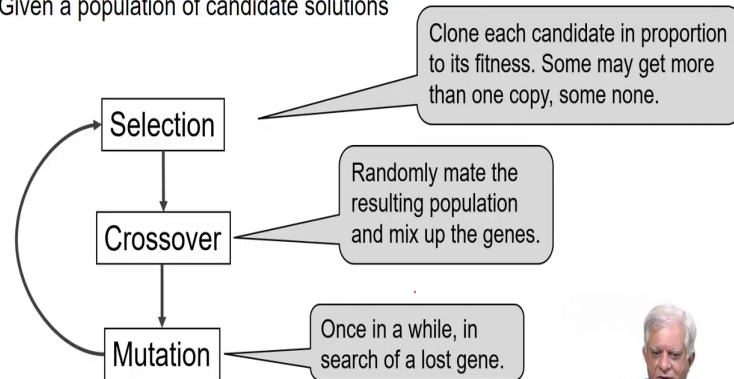
Now, in his famous book, the British evolutionary biologist Richard Dawkins, the book is called Blind Watchmaker, he empathetically argues that there is a ratchet mechanism which is built into nature. In the sense and this is what Steve Grand also said if you remember things that persist, persist; things that do not, do not.

So, the moment through some process either through the intermixing of genes or through a process of mutation, better design comes into being. Then, nature simply select keep selecting it and that survives essentially and the main argument that Richard Dawkins puts forward is in the blind watchmaker is that there is no watchmaker which has devised this clock work like universe which seem, so work so perfectly; you know the earth going around the sun and the moon going around the earth and life kind of thriving. He said that all this is simply a matter of persistence or a natural selection an evolution and this is the theory that we are also trying to kind of get inspired by.

(Refer Slide Time: 05:36)

Genetic Algorithms : Artificial Selection

Given a population of candidate solutions



So, if you remember genetic algorithm says a process in which selection is artificial as opposed to nature. In nature life forms come into being and they compete and they may or may not survive and whichever forms survive, we call them the fit; fit or we call them to be fit essentially.

Whereas, in genetic algorithms, we start with the population of candidate solutions and we first go through a process of reproduction or more popularly as it is called selection, in which we make copies of candidates which are in proportion to the fitness of each candidate.

So, of more fit candidate, more copies made and a less fit candidate may not get any copies made at all. So, this process of selection is what we do by imposing an external fitness function. So, what is this fitness function? This fitness function is basically the function that we

are trying to optimize. So, sometimes people call it as a objective function in the optimization community.

In our search algorithms, we have called it the heuristic function and we are trying to find the best value for the heuristic you see. But principle is common to all of them is that there is something that you want to optimize and that something is expressed in the fitness function and in the genetic algorithms as we implement them as they were devised by Holland and popularized by Goldberg as we said earlier.

We first do the process of selection and this is done in proportion to fitness. So, you can imagine the kind of a roller wheel which you spin N times, if the population is of size N and in after each spin, you figure out which candidate to make a copy of and you keep doing that for N times. Having created the new population or selected the fitter population from the this thing, we next do the mixing up of genes in operation which is called Crossover.

So, and the way crossover is done here in genetic algorithms is that you randomly mate the resulting population and you mix up the genes essentially and the mixing up is done through this process called crossover which had we had a brief look in the last session; but we will have a more detailed look now onwards . The last thing that we do is mutation because as we have often observed that mutation is what results in a random change.

It is like making a step random walk move in the space of possible candidates and the reason why we keep mutation is for the sake of completeness is that if we have for example, lost a particular gene, a population has lost a particular gene; then, maybe mutation may help you regain it and this is in the context of genetic algorithms.

In the real-world mutations happen once in a while and some mutations or most mutations in fact, are bad for that particular individual and particular individual does not propagate. But once in a while a mutation is good for that individual and then, the that individual prospers and thrives and propagates it is genes into its children and so on.

(Refer Slide Time: 08:58)

The Algorithm

GENETIC-ALGORITHM()

- 1 $P \leftarrow$ create N candidate solutions \triangleright initial population
- 2 repeat
- 3 compute fitness value for each member of P
- 4 $S \leftarrow$ with probability proportional to fitness value,
 randomly select N members from P *with repetition*
- 5 offspring \leftarrow partition S into two halves, and randomly mate
 and crossover members to generate N offsprings
- 6 with a low probability mutate some offsprings
- 7 replace k weakest members of P with k strongest offsprings
- 8 until some termination criteria
- 9 return the best member of P



So, here is again a recap of the algorithm; again, written in a slightly more neater fashion. So, we start off by creating a population P, which is the population of N candidate solutions, where do we get this P from? We could get it by randomly choosing candidates.

For example, if you are doing the SAT problem by randomly choosing N k bit strings, if there are k variables. All, in fact, they could be obtained by other methods. So, for example, we might to iterated hill climbing, starting from random candidates to get some reasonably fit candidates and then, on top of that we might do algorithm like genetic algorithm or another.

So, very often people combine methods. They make hybrid methods. You do first one optimization approach and then, follow up with another approach. So, this could be done here

as well. And then, we do the following that we compute the fitness for each member of P and then, with the probability which is proportional to fitness, we make copies of every element.

So, we randomly select N element members from P. So, remember the population size is N and starting with a population, initial population of N, we produce a new population of N elements and this is done by random selection and you must keep in mind that this is with repetition that you are allowed to select the same candidate again and again.

Then, from this selected candidates, we produce a set of offspring and that we do by partitioning the set S that is the fitter candidates into two halves, and randomly mate and crossover members to generate N new offsprings and from this offspring, we take the k most fit or strongest offspring and we replace the k weakest members of P which is the original population we started with, with these new members. Of course, k could be equal to N also which would mean that you are replacing the entire population with the new population.

But sometimes some researchers or people who implement things like to keep the previous best few candidates to carry forward to the next cycle and then, we do this till some termination criteria, it could be the number of cycles that you decide based on computing power available or it could be the fact that there is not much change in the fitness levels of the population.

So, for example, you could plot as the algorithm goes forward how is the average fitness or maximum fitness changing and if that is kind of becoming stable, you could terminate essentially and you may return the base member because we are basically looking for one answer.

(Refer Slide Time: 12:05)

Complexity of SAT



$$F = (a \vee \neg b) \wedge (\neg a \vee c) \wedge (\neg c \vee f) \wedge (\neg e \vee f) \wedge (\neg a \vee b) \wedge (a \vee \neg d)$$

2SAT - problems where each clause has at most two literals
- known to be in P

$$F = (a \vee \neg b \vee e) \wedge (\neg a \vee b \vee c) \wedge (\neg c \vee d \vee f) \wedge (d \vee \neg e \vee f) \wedge (\neg a \vee b \vee d) \wedge (a \vee \neg d \vee f)$$

3SAT - problems where each clause has at most three literals
- known to be NP-complete

kSAT - $k > 3$
- could be much harder

For accessing this content for free (no charge), visit : nptel.ac.in

Artificial Intelligence: Search Methods for Problem Solving

Deepak Khemani, IIT Madras



Now, one of the problems that we have been discussing is SAT and let me just reiterate what is the complexity of this SAT problem. So, we often classify SAT into different subclasses. There is this subclass called 2SAT and this 2SAT is a SAT formula in CNF. So, we very often just translate everything into CNF and CNF is conjunctive normal form, where the main connective is the and as we had discussed earlier and the what is between the ANDs is or what is connected by the ANDs are the set of clauses and clauses will have literals which are either variables Boolean variables or the negation of Boolean variable.

In 2SAT the number of clauses, the number of literals in every clauses confined to at most two and so, this example that we have here of is a formula of two 2SAT because every clause that we have has two variables in it or sometimes a negation of a variable.

Now, it has been shown that 2SAT is in fact, can be solved in polynomial time by deterministic methods and there is a very well-known algorithm called DPLL which we may visit later on when we are looking at logic again. But there are algorithms which solve them in polynomial time. On the other hand, if we increase the number of literals in a clause by 1, we get this problem which is called 3SAT and 3SAT is what was shown by Cook to be in NP-complete.

So, remember that NP-complete is those class of problems which can be solved in polynomial time provided, we have a nondeterministic machine. So, it is NP stands for non-deterministic polynomial time and these problems typically are hard to solve. So, we have spoken about complexity of SAT; how it grows exponentially and. So, if you are solving problems which are 3SAT or more complex; then, you very often cannot rely on deterministic methods.

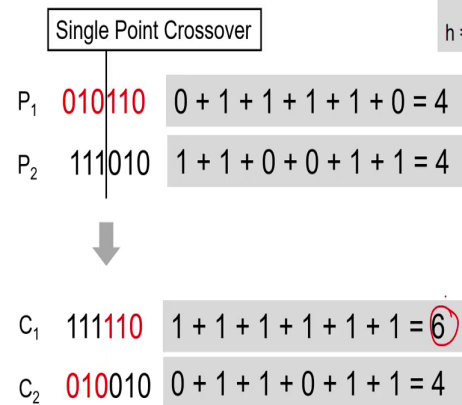
In fact, there is a whole library of SATs solvers available commercially as well and many of them or most of them rely on randomized algorithms which means that you can run that algorithm and there is a pretty good chance that if there is a solution, you will find it essentially.

The only thing is that these algorithms may not be complete that in the rare case there may be the it may be that there is a solution that you do not find; but that happens extremely rare rarely and its rare enough for you to ignore essentially. kSAT would be when k is greater than 3 and these problems could be harder essentially.

Now, I may also mention that that when you talk about Boolean formulae's or logical formulae's like this, you can always translate one formula into a different kind. So, if you have formula of k k variable, k k k literals in a clause, you can always translate it to 3SAT. Its just that the size of the formula tends to grow exponentially.

(Refer Slide Time: 15:38)

A SAT problem with 6 variables



$$F = (a \vee \neg b \vee e) \wedge (\neg a \vee b \vee c) \wedge (\neg c \vee d \vee f) \wedge (d \vee \neg e \vee f) \wedge (\neg a \vee b \vee d) \wedge (a \vee \neg d \vee f)$$

h = number of satisfied clauses



Now, it has been studied that that SAT can be quite complex ah, but let us first look at the complexity or let us first look at an example of how you could use a single point crossover to solve SAT problem. So, we have this 3SAT formula given on the top. This is the same formula; we saw in the last slide. It has got 6 clauses and each clause has 3 literals in it.

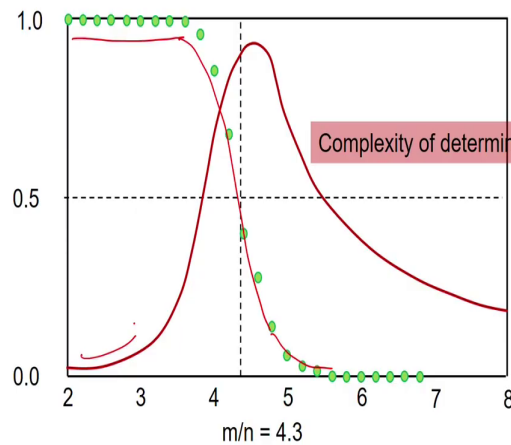
The heuristic function is the number of satisfied clauses and supposing we choose these two clauses which we have called P 1 and P 2 as the two parent clauses and then, we do single point crossover. We randomly choose one point, where we would break it up and then, we end up with the two children.

As you can see that half of the red parent is in child 1 and the other half is in child 2 and likewise for the black parent. And you can see that the two children, one of them still has a heuristic value of 4, but another has a value of 6 and if you see that there are 6 clauses; that

means, all the clauses are satisfied and this means that we have actually found a solution to the SAT problem.

(Refer Slide Time: 17:04)

Probability of 3SAT being satisfiable



m clauses
n variables



Now, people have studied the complexity of SAT and in particular we are talking about 3SAT here ah. So, remember that the 3 refers to the fact that there are 3 variables in a clause or 3 literals in a clause essentially. But in general, the number clause number of variables could vary and we let us say if there are n variables and if there are m clauses; typically, m is a multiple of n, then people have studied as to what happens with trying to solve SAT using deterministic methods.

Now, it turns out that the probability of a SAT formula being satisfiable. It is very high in the initial stages, where m by n or the ratio of the number of clauses to the number of variables is low which means there are many variables and few clauses. Obviously, you can find

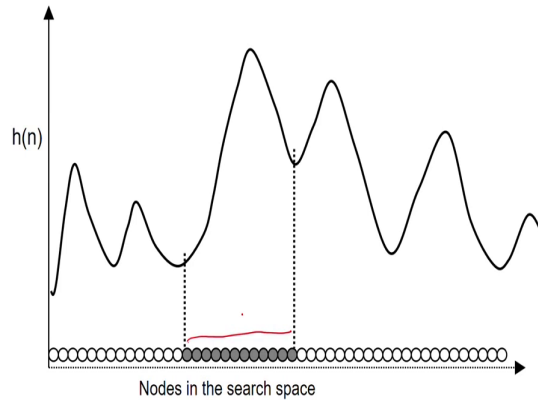
assignments to those variables which will satisfy those clauses; but at a certain point, there is a sudden phase shift and the probability of the SAT formula being satisfiable drops very rapidly and then, it tends to become almost 0.

This phase shift happens in the case of 3SAT for the value of about 4.3 which means the number of clauses is roughly 4.3 times the number of variables essentially and the second plot in this diagram which is this red colored graph shows the complexity of solving this thing. So, you can see that it is that initially the complexity is low that because you know their formulas are more likely to be satisfiable and you are more likely to find a satisfying assignment easily.

But as you head towards the transition, the complexity shoots up and later on, when the formula becomes unsatisfiable, then it is often easy to show that it is unsatisfiable. So, this is how SAT kind of complexity varies and if you are solving SAT problems with hundreds of variables, then you can see that deterministic methods may not always be suitable essentially and in fact, almost everyone uses randomized approaches to SAT.

(Refer Slide Time: 19:25)

Iterated Hill Climbing



A random starting point from any of the shaded nodes would lead to the global maximum.

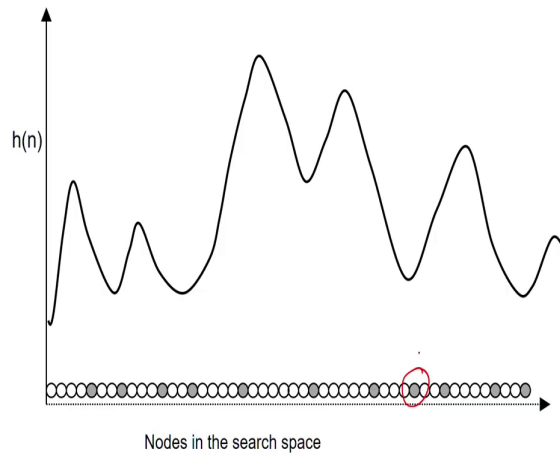


Now, how does genetic algorithms perform? So, let us re that when we talked about iterated hill climbing, we said that we will start with many random initial positions and we are talking in the context of states solution space search and we observed that there is a footprint which is defined by the heuristic function that if you happen to choose a random position in any one of these locations, then hill climbing would have taken you to the global maxima.

And therefore, if you take a sufficiently large number of candidates and the and the surfaces kind of not too bad, then you are quite likely to get solution with iterated hill climbing.

(Refer Slide Time: 20:17)

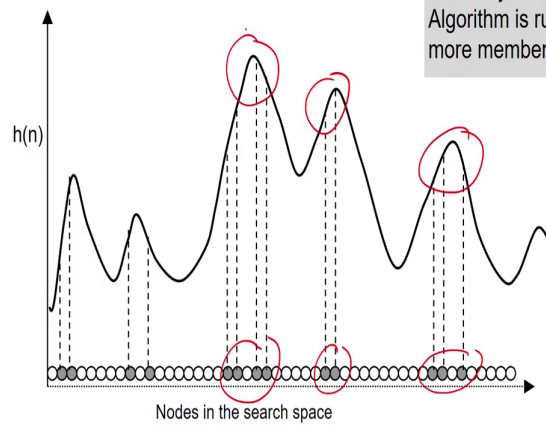
Genetic Algorithms: An initial population



Now, on a similar terrain, genetic algorithms we start off by again the random set of populations depicted by the shaded circles at the bottom here. So, for example this one and we go through a process of this selection, crossover repeatedly with an occasional mutation shown in for the sake of completeness.

(Refer Slide Time: 20:44)

GAs: The evolved population




The Initial population may be randomly distributed, but as Genetic Algorithm is run the population has more members around the peaks.



And eventually, what we find experimentally is that the population stay tend to kind of get clustered around the local maxima that happened. So, some of them would be clustered around this peak, some may be clustered around this peak, some may be clustered around the global peak and so on and in general, we find that populations become as we as we call it fitter and fitter as the algorithm proceeds and this is in some sense, mimicking the process of evolution where different populations become fitter as they compete and they get selected for the next cycle.

(Refer Slide Time: 21:21)

A Tiny Example

From: David E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, 1989 

Let the chromosome for a tiny problem be a 5-bit string

Let the fitness function $f(x)$ be the *square* of the 5-bit binary number

Let the population contain 4 candidates

We illustrate the algorithm with a single point crossover



So, let us look at a tiny example to illustrate some of the idea that we have been talking about and this example is from the book by David Goldberg the book is called Genetic Algorithms in Search, Optimization and Machine Learning and it is a nice small example and it illustrates first of all the algorithm itself and it highlights one feature that we will see towards the end, which talks about the under what conditions will this algorithm perform well. So, let us assume that we have a chromosome for a tiny problem and let the chromosome be a 5-bit string.

So, remember that the chromosome is essentially the encoding of a candidate and it is a chromosome which participates in the mixing up of genes essentially. So, remember the difference between genotype and phenotype that we had said.

Let the fitness function for this particular problem be a simple function. It is simply the square of that 5-bit binary number essentially and let us say that we have a small population of 4 candidates and we illustrate this algorithm with a single point crossover.

(Refer Slide Time: 22:40)

A Tiny Example: Selection



Initial	Binary	f(x)	Prob.	Expected	Actual	Selected
01101	13	169	0.14	0.58	1	01101
<u>11000</u>	24	576	0.49	1.97	2	11000
01000	8	64	0.06	0.22	0	11000
10011	19	361	0.31	1.23	1	10011



Total 1170
Avg. 293



So, this is the initial population that we start off with. So, you can see that there are 4 candidates shown in the 4 rows of this table that we are constructing. The first the first candidate is 01101, the second one is 11000 and so on.

Now, if you look at this candidates as binary number, then you can see that the numbers are 13, 24, 8 and 19. If you take the square of these numbers which is what we have defined to be the fitness function, then we get 169, 576, 64 and 361. You can see that the total fitness of the population is 1170 and the average fitness is 293.

Now, let us let's kind of hand simulate the process of genetic algorithm. So, remember the first step is selection and selection is done based on the in proportion to the fitness of the candidate.

So, we have 4 candidates and we have fitness values for them and if we take the fitness value of each. For example, 169 and divided by the total fitness, we will get the probability of that candidate being reproduced essentially. So, once we do that, we get the next column which is the probability of each candidate being reproduced.

And you can see that the first candidate has a probability of 0.14 not very high. The second candidate which is the fittest candidate in our population has a much higher probability of being reproduced and it is about close to 5 and likewise for this third and the fourth candidate.

So, third is the weakest amongst this and you can see that is the smallest number its square is only 64. Now, how many copies do we expect of each that is given by this column which talks about expectation. This expectation or expected number of values is simply given by multiplying the probabilities by 4 because we are going to spin that roller wheel 4 times. We have we want to produce 4 4 new candidates from this given population and let us say we spin this roller wheel, this roller wheel is basically a random process.

So, for example, you might think of this as a roller wheel and then, you would have something like this. So, you can see there are 4 sectors in this roller wheel and you spin it randomly and at some point, you have arrow sitting here and once you spin it and once it stops whichever candidate comes and stops for this that is the one that is reproduced.

So obviously, since this is a random process, it is a process anything could happen; but we let simulate the fact that you know it happens along expected lines which means that the candidates which are fitter, they are more likely to get copies made.

And supposing we do this experiment and then, we get the actual population we find that the first candidate has got 1 copy; the second candidate which is 11000 has got two copies; the third candidate has produced no copies and the fourth one has 1. So, this is the selection we have and you can see that there are 2 copies of the second candidate which was the fittest of them ok.

(Refer Slide Time: 26:23)

A Tiny Example: Single Point Crossover

Selected	Crossed-over	Binary	f(x)
01101	01100	12	144
11000	11001	25	625
crossover →			
11000	11011	27	729
10011	10000	16	256

Fitter population

Total 1754
Avg. 493



So, having done the selection phase the next phase is crossover and so, let us see. We divide the we may we randomly mate up we randomly pair the population. So, let us say that we pair the first and the second and the third and the fourth and then, we randomly choose a point for each across which we will do single point crossover.

So, let us say that these are the points that we have chosen and for doing single point crossover and this results in the crossover population which you can see that each child has

got one part from parent 1 and the other part from parent 2. Parent 1 in both the sets is red and parent 2 in both the sets is black and you can see that each of the children is a combination of red and black; split across the points that we randomly chose essentially.

So, let us look at this population. The numbers that in a new population are 12, 25, 27 and 16 and if you compute their squares which is the fitness value that we are interested in, we get these values 144, 625, 729 and 256.

Now, you can see that the average fitness has gone up from the previous cycle. In the previous cycle, it was 200 something and now we have 493. So obviously, you can see this process that by selecting fitter candidates, we can increase the average fitness of the population as well.

(Refer Slide Time: 28:02)

A Tiny Example: Cycle 2

Crossed-over	Binary	f(x)	Prob.	Expected	Actual	Selected
	01100	12	144	0.08	0.33	0
—	11001	25	625	0.36	1.42	2
⤵	11011	27	729	0.42	1.66	2
	10000	16	256	0.16	0.58	0
		Total 1754			Total 2708	
		Avg. 493	➔		Avg. 677	



A fitter population BUT less diverse



Now, with this new population assuming that there is no mutation here we go through the cycle all over again and so, we start with this crossed over population and again, we repeat this process of selection. So, if you remember these are the numbers and these are the fitness values that we just saw. So, let us say that these are the probability. As you can see in this column the probabilities of the first and the fourth candidates are pretty low and the second and the third candidates are pretty high. So, you would expect more copies of second and third to come up in the next generation. The expected values as you can see are 1.42 and 1.66 for the second and third candidates.

And let us assume that this is what really happens when we actually do the randomized generation, the roller wheel process that we get two copies of candidate 2 and two copies of candidate 4 and that gives us this new population which is as you can see two copies of candidate 1 and likewise two copies of candidate 3.

Now, if you look at this population you will and if you do the computation, you will see that the total fitness has gone up from 1754 which was the one that we just started within the second cycle and the average fitness as in fact gone up to 677. But if you look at this new population carefully, you will see that there is this one bit which is 0 in all the candidates.

And now, if you imagine this single point crossover being done with this population and subsequently, you can see that there is no way that this 0 will become 1 because all the all the candidates have 0 in the third position and it will always remain 0 in the third position and that is one of the reasons why we do mutation.

So, if you introduce the step of mutation, then hopefully in one cycle somewhere this third bit will become 1 and then, it will get selected and then, it will propagate and notice that the fittest candidate is where there are all 5 ones because that is the biggest number that you can think of and that is what we are trying to achieve even though it is this roundabout way of doing it by a genetic algorithms.

So, there is a danger that we may end up with fitter populations, but which is less diverse. Now, people say that this is true in nature as well, that sometimes certain species or sometimes even subspecies, they become so good at whatever they are doing that they become less and less diverse essentially and an example that we often give is that of the Cheetah. The Cheetah as you probably know is known to be a master hunter and known for its rapid acceleration and speed.

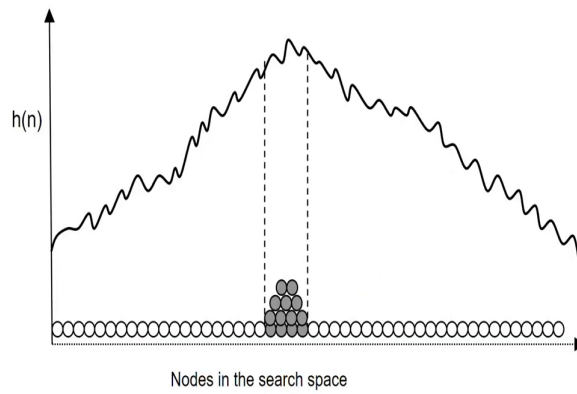
So, much so that many car advertisements, they use Cheetah as the symbol. But the problem with Cheetah is that its good at acceleration and good at speed. So, it survived very well in grasslands in Africa, where it could you know chase the prey and catch them, but with human beings changing the world and grass grasslands vanishing and forests vanishing and so on.

The habitat for creatures like Cheetah is diminishing and they are now unable to adapt to the changing environment and that happens because they have become less diverse. Most of the Cheetahs have the same genetic makeup and how much mix; however, much you mix them up, you will get the same Cheetah back essentially.

So, that is a danger with having a small population which is not diverse, that you cannot do much change after that. So, one of the lessons that we learn from this small example is that for genetic algorithms to work well, you must work with large populations large and healthy population.

(Refer Slide Time: 32:27)

The population may become less diversified



Artificial Intelligence: Search Methods for Problem Solving

Deepak Khemani, IIT Madras

So, this diagram basically depicts that under certain conditions the population may become less diversified and all may be very similar in nature. So, they maybe optimal in one scenario in the given scenario we can see in this diagram; but in a changing world, they will not be able to adapt and that is a totally different aspect of evolution.

(Refer Slide Time: 32:55)

GAs: practical matters



Candidates are encoded as chromosomes

Genes are like components in the candidates

GAs work by inheriting components from two parents

A careful encoding is needed
- we look at Gas for TSPs next

A large diverse population
is critical to for performance



So, the practical matters for genetics algorithms, as we know candidates are encoded as chromosomes and genes are like components in the candidate. So, if we can modularize our solution, if we can have it made up of different components; then, there is a possibility of mixing up components or mixing up of genes essentially and that is what we did when we did for ripple SAT, we said that every bit is a component and you can just mix up bits and so on.

In the next session, we will see that for some problems like TSP, it is not so easy to do this mixing up. But nevertheless, there are ways of doing it. So, GAs or genetic algorithms work by inheriting components from two parents and that is where the mixing up comes in and as I said just now, a careful encoding is needed and we will look at Gas for TSPs next and the last thing, we learnt was that a large diverse population is critical for performance.

So, if you have a large diverse population, there is always the chance at the fittest will emerge from them and obviously, the population should be fit as well. So, the best thing is to have a large population with high average fitness and that is when genetic algorithms perform the best.

(Refer Slide Time: 34:27)



Solving TSP using Genetic Algorithms



So, in the next session, we will look at solving TSPs and see how we can solve the Travelling Salesman problem using genetic algorithms ok.

So, see you in the next session.