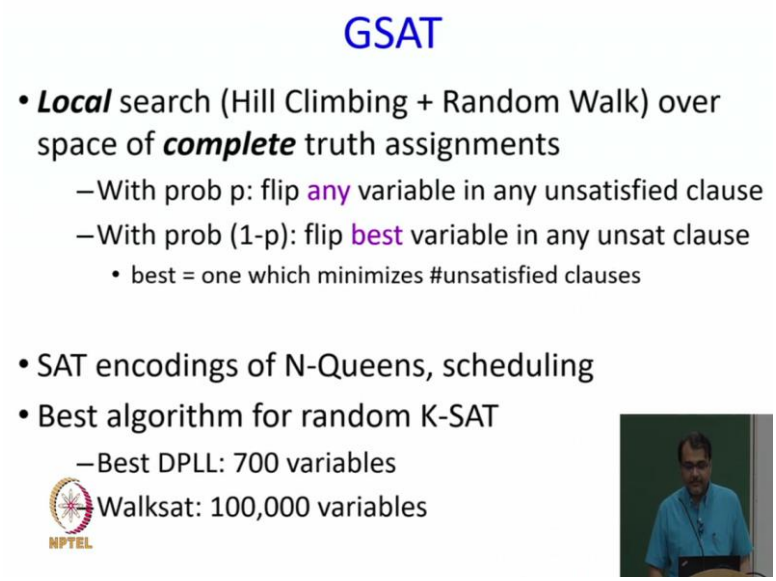
Artificial Intelligence
Prof. Mausam
Department of Computer Science and Engineering,
Indian Institute of Technology Delhi

Lecture-49
Logic in Artificial Intelligence: SAT Solvers: WalkSAT Algorithm, Part-8

Now, before we talk about something else we also always have to do with systematic search you always have to do local search that is a rule in our traditional AI local search is so dear to my heart. So, we have local search algorithms again a local search algorithm.

(Refer Slide Time: 00:36)



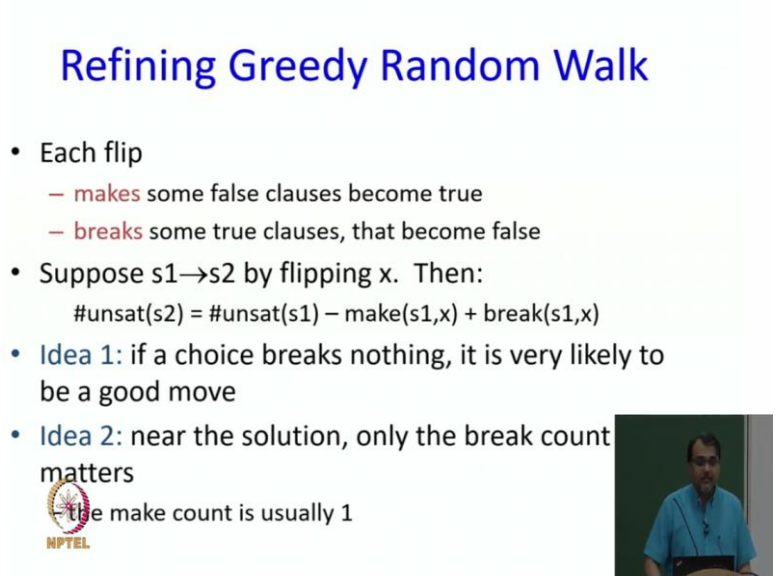
Always work in complete truth assignments. Typically, we have many such solvers, you can develop your solver it will take a full assignment, it will change some variables that would be a neighbourhood function you want to have an objective function. So, that will be how many clauses satisfied and how many clauses are unsatisfied and you will use some kind of a Climbing with random walk or hill climbing with random restart or something like that.

So with random walk, it would be something like the probability p flip any random variable in an unsatisfied clause with probability 1 - p flip the best variable in an unsatisfied clause, the best would be defined as 1, which minimises the number of unsatisfied clauses. If you have a SAT encoding of N Queens, the best DPLL algorithm can solve up to 700 variables problems.

But best walkSAT algorithm can solve over 100,000 variable problems and again these are not we have not described walkSAT, but best local search algorithm can solve upwards of

100,000 variable problems. And this particular algorithm that I just showed you is called GSAT. Now, we will also talk very briefly about the walkSAT algorithm. It is a nice algorithm.

(Refer Slide Time: 02:07)



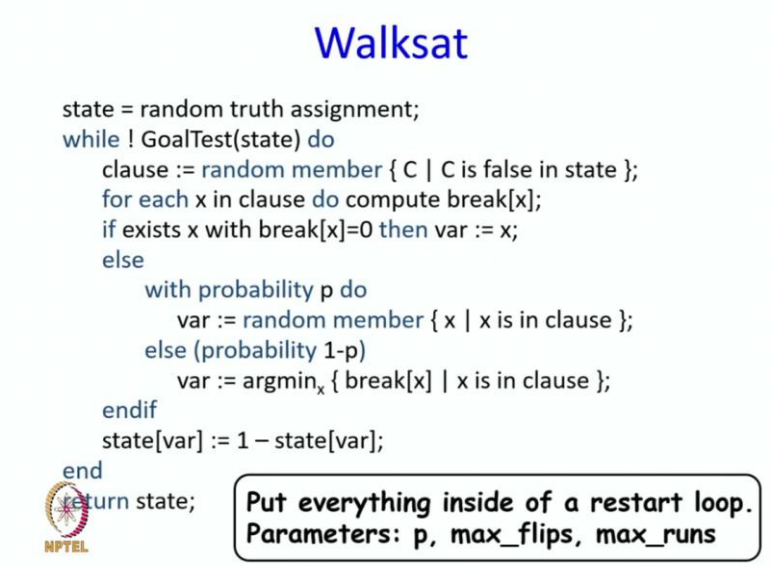
And it comes from the observation that let us say define this notion makes that I made a flip. When I made a flip, some false clause became true, and some true clauses possibly became false. The number of false clauses that became true is the makes. Function and number of true clauses that became false is the break function. And whenever we make any flip, let us say we go from a full assignment s1 to an assignment s2 by flipping x.

Then actually what has happened is the number of make s1 x some formulas have made and simple formulas have been broken. So number of unsatisfied clauses so, number of unsatisfied clauses s1 minus how many clauses it made, plus how many clauses it break everybody with me so far? Now, we will use 2 ideas to refine, GSAT said further. One idea is if a choice breaks nothing, it is a good choice that is obvious.

Even if it makes only one thing if it breaks nothing, it is a good choice in terms of greedy alignment. And then second idea comes from an observation of how these algorithms work. The observation is that when you are close to the local optimum, when you are, you know, set some variables in a specific way that you know, your number of unsatisfied clauses is small but they are tightly constrained.

Then every flip makes only one clause. Typically, if there was a flip that made many, many clauses you would have been, you would have done that flip long time back, usually. So typically make count as 1. And so the only thing that matters is the break count how many it breaks.

(Refer Slide Time: 04:14)



So the walkSAT algorithm works like this. This is sort of like a simulated annealing algorithm in simulated annealing. We had the temperature, it is not like that. But in simulated annealing, we will use to pick up variable randomly pick up move randomly. So in the same way, we will pick up clause randomly. So let us pick up random clause. But we do not pick up random satisfied clause we pick up random unsatisfied clause. So let us say we pick a random clause C, which is false in the current state.

Now for this clause, we go over all the variables in the clause. So for each x each variable x, we compute break of x we basically compute, if I flip this variable, how many clauses will it break? And now if there exists a single variable or some variable with break of x = 0 what do we do? We just flip that variable with nothing is getting broken. So, who cares you know, every only something positive is going to happen with this flip, nothing is going to break. So let us make that move.

So if there exists something with break x = 0, then variable has been x has been flipped as we walk random walk. So, we do something with probability p and we do something with probability 1 - p with probability p, we do a random move that means we pick up random

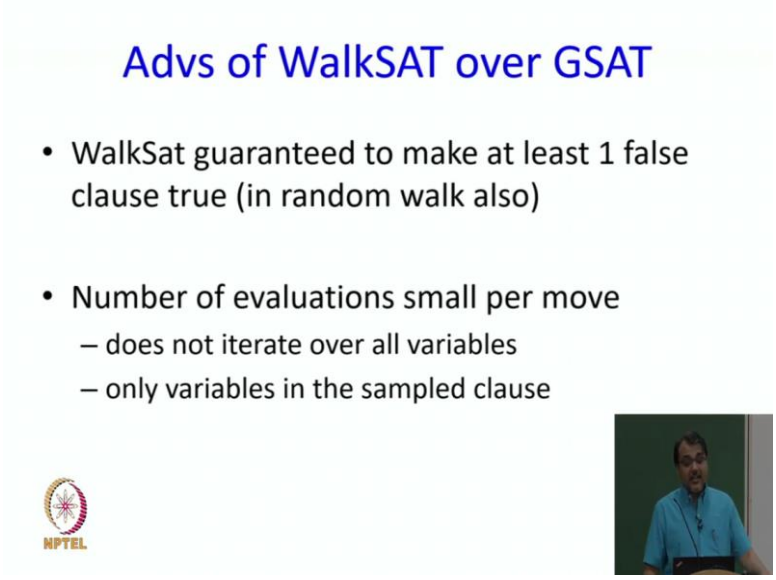
member from this x and flip it and with probability 1 - p we pick the best member x best defined with it breaks minimum things and flip that this is my walkSAT algorithm.

And of course, we can put everything inside a random restart loop. So, after some time it will say you have been searching in this hill for far too long jump to a different and it will have some parameters like how much is the probability a random walk versus greedy move? How many max flips do you do before you do a random restart how many restarts you do, etc. Now, there are some natural advantages of walkSAT over GSAT can you get?

So this is the walkSAT algorithm it first picks a clause then looks at only the variables in that clause, and then either takes a good variable and makes a move or takes a random variable And makes a move whereas GSAT did this GSAT said that with probability p I will flip some variable in an unsatisfied clause. So I will take a random clause pick any random variable and make a move with 1 - p the probability I will take the best value in any unsatisfied clause.

So I will go through all unsatisfied clauses and then I will pick the best 1. So, why might walkSAT be better? If there is no beaking then there it is. So, if nothing if there is a variable which breaks nothing in the case of walkSAT, it would do it definitely, but in the case of G stat, it will do it with probability p. That is not exactly true. Why is it not exactly true? Because first we have picked a random clause here we are not going over all unsatisfied clauses and actually that is the strength of walkSAT believe it or not.

(Refer Slide Time: 07:51)



So, the strength of walkSAT is that the number of evaluations is very small per move. Why? Because I am not going over all unsatisfied clauses and finding a variable that has the best characteristics which would be, which would be more greedy, It would be more taking a better move, but you will have to do a lot of competition. See when in doing local search, it is important that each move is done extremely fast, so that I can do a lot of search quickly.

So, because it does not iterate over all variables, it does not iterate over all unsatisfied clauses. You only iterate over variables in one sample clause, therefore, the number of evaluations is extremely small per move, possibly at the cost of the greedy step. But it is still makes one unsatisfied formula clause true, which it needs to make for all the clauses. So, it is not clear that makes a mistake or that is not a good idea.

So walkSAT is guaranteed to make at least one false clause true even in the random walkSAT and the number of evaluations is extremely small per move. So therefore, in practice, it was founded walkSAT is a much faster algorithm than GSAT. And therefore, walkSAT became the de facto standard for solving GSAT formulas using local search. Now this sort of ends your chapter on logic.