**Modified**

As you can see, the easiest and simplest way to solve the CSP problem is to use deferred search with backtracking. But if we just do normal deferred search, then that strategy is uninformed because the deferred search algorithm doesn't know what branch of the search tree it should go to. And then that can become very inefficient.

So in this example, you will see that without choosing the right branch of the search tree to go down, then backtracking can be very inefficient.

So this is the example on the previous slide where we start with the empty assignment. All the regions are not assigned any color. And then we start with WA assigning the color red for WA and after that we choose empty, assigning the color green to empty. After that we choose Queensland Q to assign a color. There are two possible colors to assign because Q is next to Nt, so it cannot be green, it can only be red or blue. So either red or blue can be for Queensland Q.

Now, let's just assume that we choose to expand this node and so we choose to assign blue to Queensland. Then the next step is let's say that we choose to assign a color for New South Wales, NSW. Now, Nisa Wells is next to Queensland. It cannot be blue. So clearly we will have to choose either green or red for Nisa Wells. Let's assume that we choose green for Nisa Wells.

And then let's say that the next step that we do is to choose a color for Victoria. Now, New South Wales has been colored green, so Victoria cannot have color green, it can only be blue or red. Then let's say that we choose to color Victoria by red and so we have red for Victoria.

Now, if we continue to expand this node and then consider Tasmania, then we can choose any color for Tasmania because it's not adjacent to any of the regions. But here you already see a problem. If the next variable to color is SA, South Australia, you can already see that there's no color for South Australia. So South Australia is adjacent to WA. It cannot be red adjacent to Nt, it cannot be green adjacent to Queensland, it cannot be blue. So that means that we will have to backtrack.

So let's say that we backtrack here and then we choose another color for Victoria. Now, we cannot use red and cannot use green either because Victoria is adjacent to Nuisance Wells and therefore it can only be blue. We choose blue to color Victoria and so we have Victoria being blue.

But then again, it is not a feasible solution either, because the problem with South Australia is not because of the color of Victoria or New South Wales, it's because all three colors, red, green, and blue, have been used. And so here it is already a bad place to continue. So instead of having to go through all this search, which can be a very big search, we could easily rule out this branch of the search by checking South Australia first and then immediately knowing that there's no possible assignment here. So that is the first thing.

So that means that there is a way to identify a good variable to consider next. So that means that between South Australia, New South Wales, and Victoria, South Australia should be considered first.

Not only that, there is also another thing that we can actually consider by looking into this. Then we may actually see in advance that there is no value that can satisfy the constraints for South Australia. So that means that we don't even have to consider the assignment for South Australia. We already know that this assignment of red to WA, green to Nt, blue to Queensland cannot really lead to any consistent assignment of variables.

This example showed that backtracking without any particular smart way to think of how to assign the value to the variables can be very inefficient. And for that reason, we want to improve backtracking efficiency. Backtracking efficiencies are a number of general-purpose methods that can give huge gains in speed for CSP solving. So that means that we are going to search for the solution in a smart way and not in an uninformed manner.

So the three main techniques for improving backtracking efficiency are to decide on which variable to assign next. As you can already see, if we choose South Australia, it's better than choosing New South Wales. Not only that, there is also when you choose a variable, then the question is among the values that can be assigned to that variable, which value should be tried first? So in what order should we try these values for that variable?

And I also mentioned that there are some inevitable failures that can be detected early. For instance, after you assign red to WA, green to Nt, and blue to Queensland Q, then we can detect the inevitable failure already without having to try New South Wales, try Victoria, try Tasmania.

So these are the three major techniques for improving backtracking efficiency. We will look at them one by one. The first one is to choose the variable to assign, then the technique is that we look for the most constrained variable. That means that we are going to choose the variable with the fewest legal moves, legal values.

Okay, so for instance, let's say that we have already done all these things. So from the no-empty assignment, we choose to assign red to WA, and after that we choose green to empty. Now, from here, among the variables that can be considered, so we have SA, we have Queensland Q, we have New South Wales, we have Victoria, we have Tasmania.

The one that is the most constrained variable is SA. So SA is the one that essentially we already see that it's adjacent to both WA and Nt. Because of that, it can only take the only value it can take is blue. On the other hand, the other variables like Queensland, it can take either blue or red. Or New South Wales it can take any colors, red, blue, green and same for Victoria, red, green, blue are possible. And Tasmania, it can also take any variable, red, green, or blue.

The one that is the most constrained is SA and that would be the one that we will choose to assign a variable to assign a value next. And in this case, there is only blue that we can assign. And you can see that if we do this, then this will be able to lead to some solution and then the solution will be somewhere at the end of this branch on the search. This technique, also known as the minimum remaining values (MRV) heuristic, because SA has the minimum remaining values, this has only one, while Queensland has two. New South Wales, Victoria, and Tasmania, they all have three. And therefore we will choose SA to assign a variable name.

So if we use the most constrained variables, then we are actually going to do the whole process with this particular technique. By doing that, we essentially will choose, for instance, SA to assign a value first, because it is adjacent to the most ordered regions. So that means that it is involved in more constraints than any other region and therefore we will find a color for it. In this case, let's say that we choose blue and then after that there are a number of variables that can be candidates for the next variable.

So we have Nt now adjacent to this one, therefore rule out blue and at the same time adjacent to the other two states and same as Queensland is adjacent to SA, therefore we rule out blue as well, and it's only adjacent to Nt and New South Wales and so on and so forth.

And so in this particular case, there are three most constrained variables: Nt, Queensland, and New South Wales. And in this particular case, we have equally most constrained variables and therefore we use a tiebreaker. We choose one of them randomly and in this case, we actually choose Nt. We assign a color green for this one and then the next one is this one because it has only one color that it can take, in this case red. And therefore we will choose Queensland to assign it and with Queensland, we assign red and then we can continue the process.

By doing this, we actually reduce a lot of backtracking. Another technique that has been used very efficiently is least constraint in value. So opposite to the variable, when after we already have a variable, so knowing that we want to assign a value to this variable, there are a number of legitimate values that you can assign to this variable. Then the heuristic tells you to choose the least constraint in value.

The least constraint in value is the one that rules out the fewest values in the remaining variables. And so let's just take the example of this Map Coloring problem again. So after this process, now from this node, let's say that you happen to choose Queensland to assign a value next. So with Queensland, there are two possible colors that you can color it because it's next to Nt. So it cannot be green, it can only be red or blue.

So with red, then if you look into the other variables, the remaining variables, then it allows one value for SA. But if you choose blue, it allows zero values for SA. And because of that, we choose the value such that it has the least constraining variable. So we are not going to color Queensland by blue, okay? We color it by red in order to allow the maximum number of values for SA.

So these two heuristics alone, the least constraint in variable and the least constraint in value and the most constrained variable combined together can make the 1000 queens feasible, versus when we don't really use these heuristics, and then we can only deal with 25 queens. So you can see the huge improvement in terms of the ability to find a solution. So without the heuristics, we can only deal with around 25 queens. With the heuristics, we are dealing with 1000 queens.

And then the next technique that I already mentioned is that you want to identify the assignments that lead to inevitable failure and then try to avoid going down that path. Okay? The idea is that we try to keep track of the remaining legal values for the unassigned variables. That means that we add a new data structure to our algorithm. This data structure will keep track of the remaining legal values for all unassigned variables.

When we start with the empty assignment, so no variable has been assigned a value, then clearly all variables can take any of the values. So, WA is allowed for all three colors: red, blue, green. Same for Nt. Same for Queensland. Same for New South Wales. Same for Victoria. Same for SA. Same for Tasmania.

Now, let's say that our search starts with WA and then assign red to WA. Now, WA is already red. So this data structure that keeps track of remaining legal values for unassigned variables will say that there are two states that are adjacent to WA: Nt and SA. So both of them cannot be red. So Nt can only be green and blue and the same as SA, it can only be green and blue.

And then let's say that the next one that we assign is Queensland. And then we assign green to Queensland, okay? And so again, Queensland is adjacent to Nt and SA. And because of that, both Nt and SA no longer can be green. And so we eliminate green from Nt and SA. So Nt can only be blue and SA can only be blue.

Now, at this point we could already see the potential issue because a quick check will be able to tell us that Nt and SA cannot have the same colors because they are next to each other. And now we only have the blue color to give to both of these states, both of these regions, Nt and SA. And that's not legitimate. And because of that, we could already say that Queensland cannot be green and therefore we have to try a different branch of the search tree.

On the other hand, if we continue to search, then let's say that we choose Victoria to assign next. Then we happen to choose blue to assign to Victoria. And we can already see that now Victoria is adjacent to SA and because SA could only take blue, now you ruled that out as well. And so there is no color that is allowed for SA.

By doing this, using this additional data structure, we are able to say that this assignment is not feasible because there are no values for one of the variables and therefore we can start backtracking from here. So you can see that this data structure allows us to identify the inevitable failure early on rather than continue with the next variable. For instance, trying to assign some value to Tasmania or trying to assign some value to New South Wales.

So we could already identify the issue early on, but as I said, actually from the previous slide, we don't even need to go to Victoria and assign some value to Victoria. We could already identify that because both Nt and SA could only take color blue and then so they are adjacent to each other and therefore this node is already not part of an assignment of a consistent solution.

**Summarise**

The text discusses solving Constraint Satisfaction Problems (CSP) using deferred search with backtracking. In an example involving a map coloring problem, the inefficiency of uninformed search strategies is highlighted. To improve efficiency, three main techniques are introduced:

1. **Most Constrained Variable:** This technique focuses on choosing the variable with the fewest legal values to assign next. By selecting the variable involved in the most constraints, the search process becomes more targeted.
2. **Least Constraining Value:** When assigning a value to a variable, this technique prioritizes the least constraining value – the one that rules out the fewest options for other variables. This approach helps avoid early dead ends.
3. **Inevitable Failure Detection:** A data structure is introduced to track remaining legal values for unassigned variables. This structure helps identify potential issues early on, even before fully exploring a branch. It prevents pursuing solutions that are bound to fail.

Using these techniques, significant efficiency improvements are achieved in solving CSPs. For instance, applying these methods enabled solving a 1000-queens problem, which would be challenging without these heuristics. In contrast, uninformed approaches struggle with even smaller problems like the 25-queens scenario.

***Important***

**CSP ALGORITHM**

The easiest and simplest way to solve the CSP problem is to use defer search with back checking. Back checking without any particular smart way to think of how to assign the value to the variables can be very inefficient. Three main techniques for improving budgeting efficiency are to decide on which variable to assign next.

**ANTIPLYING THE HEURISTIC**

With the heuristic, we are dealing with 1000 queens. The next technique is that you want to identify the assignments that lead to inevitable failure and then try to avoid going down that path. The idea is that we add a new data structure to our algorithm. This data structure will keep check of the remaining legal values for all un ASSigned variables.

**Original**

So in the next few slides, I'm going to cover one of the important concepts in AI to allow you to view a number of important intelligent systems. These are the concept of intelligent agents. So the idea of intelligent agent is that they are operating within some environment and then the agents are trying to achieve some design objectives. So this design objective allows the agent to know what it would like to achieve when operating within this environment. These design objectives will be encoded in the performance measure that the agent had access to. And of course the agents will try to achieve this performance measure with the best possible actions they can perform in this environment. For the agent to do that is also contained the sensors to allow the agents to observe the state of the environment based on those observations of the state of the environment and know about the performance measure is trying to maximize the agent using its intelligence by having the ability to reason to perform inference and perhaps also learning as well. The agent will combine the information about the environment with its reasoning capability and knowing what it try to achieve in this environment, it decide on what action theater will take and those decisions of the action to take were sent to the actuators to allow the agents actuators to perform these actions in this environment. And the actions will make changes to the environment to allow the state of the environments to be updated. And then eventually the environments will get into a state that satisfy the objectives of the agents. And so all of these decisions that the agent made will be performed autonomously. So that means that the agents has full autonomy without needing some control from a humans or from any external parties. So no one needs to tell the agent what to do. And the agent simply operates within the environment all by itself, observing the environment, making the decisions and then performing the actions in this environment. So an example of such intelligent agents is the self driving cars. With the self driving cars, then these cars will need to be ridden in an environment which consists of the urban streets, the freeways, and, of course, together with many other kinds of traffic, such as all the cars, the motorbikes, could be some bicycles, could be a lot of pedestrians, and then in different weather conditions and carrying different kinds of customers and so on and so forth. So that is the environment that the self driving car agent need to operate within. These cars will be written in such a way that they maximize the performance measure. For instance, they need to ensure safety because there's no point in driving a car and then it's unsafe. And then they need to drive safely to the destination because clearly when people get into a car they want to go from point A to point B. And the important thing is the agent will eventually get them to point B. And of course in the process the car has to also obey the law. So it is written lawfully and clearly, also make sure that it provide the passenger in the cars with a comfortable ride. So knowing about these performance measures, now the agents will use its sensors and the sensors could be having the camera in order to capture the video of the environment around it, the sonar sensor, the speedometer, the laser sensor and so on and so forth in order to get information about the current environment. So with that information about the current environment, the agent will use its inference capability in order to decide on among a number of action that it can perform, which action is the best one to allow it to drive the car safely to the destinations in a lawful way and providing passenger comfort. And so those actions could be to accelerate the car, or for instance, in order to safely avoiding collisions with another car. So it need to stop the car by braking and of course, if it need to change the lane or it need to turn left or turn right when it's reaching the destination, trying to reach the destination, they need to steer the car accordingly as well. And all the sort of actuators, for instance the horn indicator to allow the car to communicate with the order traffic is also maybe used as part of the actions that the self driving car agent can perform. So that's why you can see that when we design agents, we typically need this information about the performance measure P here and then the environment E here. And of course we also need the actuators that the agent can use to perform its actions a and finally the sensor the agent use in order to sense the environment. So this form the Peas design of an Intelligent Agent okay, so now that you have these basic components of an intelligent agent, then how can you actually design an intelligent agent to solve a particular problem in a specific problem domain? Then depending on the type of problem you may consider different agent types and different agent types may give you different capabilities of these agents. And so for instance, you may have a simple reflex agents, a model based reflex agents or goal based agent or utility based agent. And then if you combine each of the above four simple agent type then with the learning capability, then you have four advanced type of agent as well. So you can have simple reflex agent with learning, model based reflex agent with learning, goal based agent with learning and utility based agent with learning as well. So let us go very quickly through these different types. So you can see that the simple reflex agents will have the form of the agent receive information about the environment through its sensors, then now it will be able to interpret the state of the environment through those perceptions from the sensors. And then with this understanding of the state of the environment, the agents already have some rules being encoded within the agents program to allow it to have these kind of condition action rules. And the idea here is that the agent only need to match the condition with the current state of the world and see which rule is applicable, which rule can be fired, and then the applicable rules will be selected. And then the actions will be produced from these rules and sent to the actuator so that the agents can perform the selected actions in the environment. So essentially simple reflex agents use the rules and it match the condition of the rules with the state of the word and the rules that is applicable then will produce the actions and the agents just perform the action. So you can see that is a very simple agent type and most of the intelligence of the agent is already encoded in these rules that the agent designers give the agents. Okay? Now that type of agents has a big disadvantage is that it can only act on the current state of the environment and it's remembered nothing about the past that it has been through and then saw certain things in some other places. Now for the agent to also use that information when choosing the decision, then the agent will need a model that store in its internal memory. Okay? So this model will store the information about the state of the environment that the agents have seen so far and then it's know about how the world evolves, the dynamic of the world and then it know about the effects of its action as well. So what my action do to change the word and together among these three things then the agents will now be able to use the information from the sensors in order to determine the current state of the word. Not only the part of the word it can observe using this sensor, but also the part of the word that it may not be currently observed, but it observed before, it observed in the last ten timestamp, in the last 20 times steps. So it incorporates those information in its state and then using its reasoning in order to construct this picture of the state of the world in the path where it can observe and also the path when it cannot currently observe. But it saw it observed those paths before. And then after that, with this more comprehensive understanding of the state of the world, again, the agent use the condition action rules in order to match the current state of the world with the conditions of the rules and then choose the applicable rules and then check out the actions to send to the actuators so that the agent can perform those action in the environment and make changes to the environment. And hopefully the changes are actually getting the agents closer to its desired objectives. All right, so again this is interesting. But then the question that you will probably pose is that even though the agents may be recognized the state of the world using its sensors and also performing some of these reasoning using its internal memory, storing the information about state of the work that it had seen previously. But then the intelligence of its actions mainly are still within those condition. Action rules that the agent designer actually gave the agents. So for the agents to be even more powerful now, instead of giving those condition action rules now we only need to give the agents the goals what you need to achieve, okay? If it's managed to get the environment into a state that satisfy the goals, then it's good. But if it is still not in a state that satisfy the goal, then it had to try to choose the action in order to satisfy these goals. And so now the agents has a lot more reasoning capability. Not only that, it gives this state internal state and then the information about how the world evolved and then information about what my action do in order to update the state of the world together with the information collected from the census. But it also able to talk about among the actions that I can perform, which action produce, which effect, and looking at the effects that my action can produce, which one allow me to get closer to the goals and then maybe the agent will even have to do some look ahead planning in order to say that if I do this action now and then after that, I do another action later and then another action later and so on and so forth, then those sequence of action will allow me to achieve my goals and having that ability to reason about the state of the world and the goals and then the sequence of action that it can perform now, the agents will be able to say that the action, the optimal action I should do now so that eventually I can achieve my goals is Action ABC and send Action ABC to the actuators to perform in the environment. Finally, you can also criticize the goal based agent that even though it has the goal and the ability to reason about the sequence of action that it should perform, it should perform to bring the environment, the state of the environment to satisfy those goals. But then that's still very much binary. A state of the environment can be either go or non go. But a lot of the situations that is not the case. Some state of the environment is better even though they are not the ideal one, but they are still better than many other states of the environment. And in particular this is the issue when you are dealing with multiple objectives such as the intelligent agents who drive a self driving car. If you ride the self driving car, then you have the objective of being safe, being fast, getting to the destination quickly, getting to the right destination, being comfortable. They are all objectives, but then they may be conflicting. If you want to be fast, maybe you have to speed in, maybe you have to run the yellow light. Now if you do that, then you violate data, the agent being lawful and maybe it's actually increase the non safety as well. Reduce the safety, right? And so there are many different objectives and they may be conflicting. Now what would be the state of the work that satisfy most of the objective? In particular, the more important one, for instance, the most important one is safety. You want to make sure that it's safe, but then at the same time if the difference in safety is very tiny, then maybe you also want to achieve a better fuel economy and then reaching the destination in a shorter time, right? And so all of these need to be encoded in this so called utility function. And the utility function is similar to the Go. But instead of saying that the state of the environment is either go or non Go, here the utility actually have the entire spectrum of desirability for the state of the environment. And then depending on the desirability, now the agent will be able to say that among these states that my actions can achieve, which state makes me happiest, and then the actions that allow the agents to get to the happiest state. So those are the states with the highest utility values. Those actions will be selected and sent to the actuators for the agent to perform in the environment. And then finally, here is the situations when you actually have the learning capability. Couple with the performance element. Here, the performance element is one of those four architectures that I presented in the previous four slides. So the performance element could be the simple reflex architecture, it could be the model based reflect architecture, it could be the Go based architecture, it could be the utility based agent architecture. Okay? And now having one of those four architectures encoded in this performance element, the agents will be equipped with the Sjar learning components. So there will be a critic to allow the agents to compare what it can observe from the environment with what it desire to achieve. So knowing that the environment is not in its desired state is able to critique the decision that it made before and then saying that I make those decisions expecting that it would give me a good outcome. But the current state of the environment showing that the outcome is not that good. So that means that my understanding about that action is wrong and therefore I need to correct my understanding of those actions. So the critics will send those feedback to the learning element to allow this learning element to make the update to the performance element. And so before I have the wrong understanding of my actions, and therefore now after I see that the state of the environment is not the same as what I expected, therefore that wrong understanding need to be corrected. And then another component is the problem generator. Because imagine that if you don't have the problem generator, then there will be parts of the environment that you have no knowledge about, then you probably will never go there. But the problem generator allow you to actually set in a goals, set in a high utility for the action of exploring the parts of the environment that you have never seen. And so the problem generator allow you to actually achieve those goals and those utility and performing the actions to solve those problems of exploring the environment. Finally, in this last slide to summarize our lecture today, I would like to remind you of a number of important concepts in AI. For instance, the four main paradigms of AIS. When we classify AI into system that think rationally or think like a human, versus system that act rationally or the system that act like a human. And how you choose to view the AI system will define the appropriate techniques. So if you choose to view system that act like a human, then maybe you need to look into some techniques. But if you want to deal with a system that acts rationally, then you may have to consider some other techniques. Then we looking at some main characteristics of intelligent systems. The idea here is that as long as we develop a system with some of these characteristics, then some of the existing AI techniques will allow you to develop those system with those characteristics. And that means that you develop an intelligent system. Okay? So these definitions allow you to have a very broad definition of intelligence systems because whenever you employ one of these AI techniques, then the resulting system is unintelligent systems, even though it may not satisfy the definitions of this system that think rationally or think like a human, or act rationally or act like a human. And so that encompass many potential systems and many potential techniques that you can use. And finally, we still very much interested in a proper AI systems. We call them intelligent agents because these are the systems that act rationally. And then we discuss about the four basic agent times. The simple reflex agents, the model based reflex agents, the goal based agent, and the utility based agent. So these are the four basic agent types. And then if we couple one of those basic agent types with learning capability, then we have a new advanced agent type. So then that means that we can couple simple reflex agents with learning and then we will have a new advanced agent type, simple reflex agent with learning or utility based agent with learning, or go based agent with learning, or model based reflex agent with learning thank you for your attention, and I see you all next week.