And they'll put you here. Oh, no no no sorry. No no no no no. This is LP two, LP three and four. Because again we have to branch X one. So this branch will be x one less than or equal to 1X1. I'm just a count of two. One? Exactly. Sorry, guys. This is LP one we can branch. But you'd be different. Yes. Sorry. So in this case, let's remind you here LP zero v branch. Which one? Here. Yeah. No. True. And this one. That's one less than one. One. One. Yes. Yes. Yes. Okay. Now we are keeping track. Okay. This is the relaxation linear luxation plus the constraint x two Greatrillioneporting two LP three Linearisation plus this constraint plus this constraint okay. Yeah. Obviously we could we could determine let's assume five and six from here in this case would be. X one less than or equal to two or greater than or equal to. Perfect. Okay, great. Let's see the results. LP three. This is LP three is correct. X two greater two two. Great. X one. Less than one. Perfect solution X11X true. Fraction. Again. Yes, I know that's the feeling object function 12 point. When we solve LP for. Infeasible. Okay LP for infeasible bound. Oh, sorry. Feasibility bound by feasibility. Okay, guys, we cannot further branch. Obviously, we cannot further branch because there is no solution to branch on is it's infeasible. Okay. Help before it's infeasible. Great. So I don't need to care about this problem. But LP again. We have here the fractional solution. So from LP you have LP four. So now you have. LP. Five is a correctional. Oh, no no no no. Jesus Christ. It's here. Right. Yes. How am I? God, yes. And. Great. Okay. LP five. You be this problem plus. Do the math. And. Yeah. Is correct. Yes. Yes. Yes. Yay! Okay, so LP three, LP four. Let's Make sure LP five LP five. Yes. Perfect x two. Less than two LP 6X2. Greater than equal to three. Perfect. Just solve LP five solution X11X22. Yay! Integer optimal value 11. Does it mean this is the optimal solution? No, but the integer if it's the integer. Now things start to get a little bit more interesting because it's the first integer. Solution when this happens is like an event in integer programming in French. About. So this solution now there is a different name. It's an incumbent solution. Incumbent solution is the best solution so far. It doesn't mean the solution is optimal. Eventually the solution is optimal. But what we know for sure you're not here. We are still here. Okay, we didn't solve this. So in terms of solution methods, LP five is the best we got. And the solution is one two with optimal value 11. Now we finally update the lower bound. Now we know in the worst case scenario our solution our optimal solution will be 11. Okay. But it turns out that we solve LP six. Okay, so this solution k is integer. So here there is no more branch. Why? Everything is integer. There is no way to keep branching LP five. Great. So it's bounded by integrity or in this case, feasibility two. In this case, LP six. We solved. Again. We solved. It is an X10 and x to three with the optimal value 12. Does this matter? No. So the lower bound is update now to 12. Because you don't know if it's optimal. What you think about. How do you feel about this? 12. Do you think we can get a solution better than 12? Okay. Do you remember the value of the linear luxation 12 point? 75. You know 75. The best we can get. This is the best you can get. It's 12. And now it doesn't mean that eventually if you branch the remaining nodes, for example, you could branch LP one just to make sure you can do that at home. You can branch LP one, but definitely there is no way to find a solution better than 12. So eventually you have a case of multiple options. So those should be the same. Exactly a different value for x one and x two eventually, but maximum 12 we cannot get anything better than 12 okay. So this is I don't know if this is right actually. Yeah, I think so. So we started like that route solution 12.75 to notes I never explored this one okay I did next. Basically I didn't need to because I knew basically the best solution was 12 and I'll be 10 or 12 solution because I was like this, okay, this like this direction. But obviously I could I could have branched before this LP one. But definitely we know the best we could have here is 11 because this is, in the end of the day, a relaxation of the original problem. Right. So this is what we call the branch inbound three. It is a very obviously for very small problems, two three variables and even not even three variables, two variables with a couple of constraints. Because obviously, as you can see in this case how many LPs we ended up solving. One. 23456. Very simple guys. Two variables. Three constraints. Let's put it this way one constraint and two constraints that basically are bound for the initial decision variables. Even this very simple and simplified problem. In the end of the day, we had some six LPs out of it. Imagine we have a real LP with hundreds or thousands of billions of decision variables. So the understand that when they show less lecture, it was less lecture that showed the gaps improving iteratively. So basically the gaps refer to this this dance between lower and upper bounds. In this case obviously we managed to find the optimal solution because it was the last node to be exploited. And we killed all the nodes, or we bound the other nodes or we pruned other nodes. Okay. There are different methodologies, but in the end of the day, the main idea is why this is an intelligent enumeration. Because if it's not intelligent, I would need to enumerate all potential. Solutions. It means that branching all the potential nodes. And for this case, for example, I didn't need to branch this one. So this is the intelligent part of this enumeration, not exhaustive enumeration, because exactly of the bounding procedure that eventually because I know or I am the lower and upper bounds, and I know there is no promises of a good solution out of a given node, and I can prune or kill or terminate a given exploration, which happens in this case because the obvious are the obvious in this case. Bounds 11.5. Considering we had promises of better lower bounds. Okay. Though. Yes. If we have. A I my instead we. 1130 20. So you sorry you got 11. Where 11.0. We got 13.5. Though we still branched 13. I know it's about 12 points and an example. Are. Okay on one side and a lower integer amount on the other side when the brands. Well, in this case, for example, if you be taking something better, for example, you should explore these nodes that give basically a better or a promise of a better solution. It's a better relaxation, so eventually you could get a better solution out of this node. Obviously 13 this case would be impossible because we know the best solution of value would be 12. But yes, if it's like for example 11.8 or something like that, he has you'd go for this, for this node and start exploring this node. Eventually this could be feasible. So if this is the case, you'd need to explore the other node and you get an optimal solution. And so we know that the highest possible amount is 12.75. Isn't it logical to directly come to 12? Because since this is a maximisation problem and 12 is the nearest. This is the nearest. What you can do is you can use a 12 as a bound, but you don't know if the optimal solution will be exactly 12. That's the point. The rounding one can give us some intuition on the optimum, on the optimal solution, but you are not sure. Eventually. This is a very simple problem, but not a problem is year round. But remember the rounding exercise? We round the solution with different techniques and we didn't get any near of the integer optimal solution. So what people in general do when you have more information about the problem structure, you can use rounding procedures to start a period in the lower in the upper bounds. We didn't do that this case, but yes, we could put it as an upper bound. But in this case the upper bound. It's 12. But you don't know if this actually is feasible for your problem. You don't know that because of the constraints you can get 12 eventually. You cannot get 12, you can get 11. So that's the point. But this would be a bound okay. You know, you raise your hands. So watch brains in branch and boundary first. Let's formalise just some terminology. Branch refers to the partitioning process of the solution space. So we have an LP have a fractional solution. We choose the fractional solution to to start branching on. And we start partitioning partitioning partitioning the solution space. The branching process may be viewed as successively finer and finer subdivision of the feasible region, where each subset is a given partition represents a problem. Why this is a partition? Because, as you notice here, a partition because there is no solution that will satisfy both. Constraints that are added simultaneously. Okay, that's why this is a partition. The solution will be either here or there. In this case, because you saw there is no intersection okay. Between these two intervals. That's why this is a partition in general, not in general. The concept of partition is the intersection of the partition is empty set. There is nothing okay. The intersection the branching process may also be viewed as a tree. The branch approach three where the root node represents the linear luxation. Okay, here are the LP relaxation of the original IP integer program formulation, and each node represents a potential sub problem in which refers to the original constraints plus the other constraints when we partition the space. In general how to implement the branching. So let's assume we have a given fractional solution or not. Integral solution x I in which x I. Sorry, this should be I q in which x I is between I one and I two. So basically this I one and I two are two consecutive non-negative integers. So for example as we saw here. We had 1.5, so 1.5 we have one and you have it to two consecutive integer numbers. All right where I want. And I show our consecutive non-negative integers. Then two new loops are created by augmenting the continuous approximation program with either the constraint x one less than or equal to I want or x I greater than or equal to I two. This is the general form okay. What? What of what we just done for the branching bounds? The branching process has the effect of shrinking the feasible region. What's shrinking. Because basically we have a given. So we have this given a feasible region, what happens when we start including these constraints or these bounds shrinking? We are. Including bounds and constraints. So we are shrinking. Okay. Making smaller this feasible region okay. But in this case interestingly when we do that we are getting rid of what integer or fractional solutions. Exactly. We are getting rid of fractional solutions. This is the main takeaway okay. We are not getting rid of any integer solutions when we do that for obviously given each branch. Okay. What else. In a way that eliminates from further consideration the current non integral or fractional solution for x I or the variable that is being printed but is true preserves all the integer solutions. Okay, obviously of the original problem. And the main thing is, which is very interesting, this process was proven to converge because we have a limited feasible region. If there is a feasible solution, the process will terminate in a couple of iterations. Obviously this can be very, very time consuming, but theoretically one day it terminates the algorithm because of the the finite number of corner points of the feasible region or optimal integer solutions anyways. In terms of bounds of the branch. Branching bounds. The bound refers to the bounds scheme that basically is used to eliminate from further consideration a given node. So we have basically three types of. Don't talk about that. We have three times of bonding procedure ruined by invisibility. That note that basically we solved and the solution was infeasible or basically infeasible. There is nothing to do. We prune or we kill, or you terminate these nodes because of feasibility. LP solution is all integer. In this case, we discard such nodes and say that this pruning. You remember the first one? The first interior solution 11 there is nothing to branch the solutions in this. This can be optimal. Okay. And when we know, for example, that one, when we know that this node cannot get a better solution because we had here better solution 12. So we don't need to process this. Not any longer. Right. Because of the boundary. Because we know this one, the LP relaxation, the LP value was 11.5. Best case scenario, the optimal solution would be 11. So there is nothing we can do with 11 considering we have already 12. So we can prune in this case by bounds. Okay. Because it was found a better bound in this case obviously a better bound with an integral or an integer solution. In terms of some very, some general rules. So when you have an LP solution and you have several LP solutions, you're dealing with a large scale problem, several fractional solutions. And the next idea is we have to start branching a given variable. So common rules. But this is not an exhaustive list. You have way way more. Okay. So variables with fractional values close to 0.5 is one rule. This is the rule that we used actually in our problem variable with highest impact on object function. This can be a bit controversial at best in practice. For example if you have in this case. We are maximising x one plus ten x two. What is the highest impact in this case? Variable. X two why we are maximising and the coefficient is way greater than the coefficient of the first decision variable. Again, there is no proof that necessarily the method will converge faster or better if you choose those rules. But again, there are different rules. And what we have is a default. In most commercial solvers, such as the simplex, we have the standard, the default ah configuration. And the standard configuration actually is a mix of different rules. Okay. There is no single recipe that is always used. There is a pre-processing first and the pre-processing that in most commercial servers we have this before finding the LP. Relaxation is a pre-processing phase. We are not discussing pre-processing, but it's very common and the pre-processing is to try to get rid of redundancy. This is the first thing, so you don't need to worry right now, like in today's world. Okay, maybe our real world problem has some redundant constraints. For example, we don't need to be concerned about that because the pre-processing, if you have a number, if this is really something to not overlook the message you will take care of. Okay. Getting rid of redundant constraints and to have a better formulation to start with. With this better formulation, we start using the branching bounds. So with this better formulation we will start the corresponding LP relaxation and we restart the message. That's why I don't know if you remember the the problem that solve the production planning problem. Do you mean by the production planning problem? We solve the integer formulation, the LP relaxation, and we use a. Please check. Remember that? Yes, yes, we did that two weeks ago. For those who don't remember anyways, do remember at some point we analyse the bounds, the gaps, the gaps of those solutions and they were like terrible batch, very bad. This means something, so probably would be nice to investigate better formulations for the linear relaxation model. Because if we start here with a very poor linear luxation, when I say very poor linear relaxation, I mean an optimal an optimal value for the linear luxation that is theoretically very far away from the original optimal solution. Okay, very far, very big gap. But see, we start branching variables based on the information this LP relaxation is giving us. If this information is rubbish, what do you can you expect the message will take a long time or a longer longer time to find a good solution. And I'm not even mentioning optimal solution because for large scale problems, we not necessarily need to find the optimal solution. We need to find a solution whose quality is okay, whose quality is good. In general, I don't know, there is no rule, but you say within 5% of optimality, that is when lower bounds and upper bounds are eventually 5% from each other. If this is a large scale problem is crazy problem, not your case. So I want optimal solution. I want 0.00 gap okay. Make sure you can find these optimal solution. Because all of you guys you'll be solving an IP formulation at least one MIP or IP formulation. So it's very important to set correctly the relative optimality gap. Otherwise you may feel you are actually solving in analysing the optimal solution. But you're not. You are analysing a bound okay feasible but not the best one. All right. So this is very important. So anyways so we have this criteria in terms of how to ah explore for example unproved nodes. Unproven nodes for example would be this node. Let's assume at a given iteration I have several nodes that I could explore why I'm talking about a bigger problem okay, bigger problem would give rise to different eventually LPs a number of LPs. So the main idea is and again here, more nodes and so on and so forth. At some point we need to identify how to keep exploiting the exploiting these these nodes. And you have different ways because eventually you have to choose between the first created node or I don't know, the the less sorry, the less created not or the first one. So you have you have to choose between the most recent one or the last one. Okay. And so we have a different rules here. So we have what we call which is the most common rule. That first search strategy. That's basically the idea is to choose an active node. An active node is the node that can be explored. Okay. It's two among the most recent descendants okay. This strategy is typically used to find a primal bound or feasible solution as quickly as possible. So if you have problems with feasibility, it's very difficult to find a feasible solution because you can have two problems. It's very difficult to find a feasible solution to your problem. Second, it's very easy to find a feasible solution, but it's very difficult to prove that this solution is optimal. This is a different problem. Okay, obviously eventually have both. It's anyways, if your problem is to find a primal feasible solution as quick as possible so as to update more quickly the bounds. So you should use depth first search strategy okay. But we have other strategies. For example best node first strategy consists of choosing the active node that is a node to further explore with the best dual bound. This strategy is typically used to reduce the number of nodes to be evaluated in the enumeration tree. So let's assume we know that the number of nodes is prohibitive. So you want to reduce the number of nodes to be explored. If this is the case, the idea is choose as the strategy for the next node to be explored, the best node for strategy. And when they say best dual bound is exactly what you are thinking, this is the case in which you. Can you find the duo? Solution based on the dual variables and with this information. Hence, for all the nodes, you can finally choose which nodes which are which, not which nodes to explore. What we have. As I told you before, what we have in practice the hybrid strategy. So if you if you read, for example, the the simplex default or good or all the solvers that are commonly used to solve LPs and MIPs, what we have is something that the solver identifies the best strategy when it starts solving a given a given problem. So this solves the solver. Start solving a given a given problem. It's it's being very difficult to find the initial solution. So let's go for that first okay I found the solution. Now let's let's now switch to the best node first strategy. So there is no unique key strategy that will be kept during the algorithm process okay. It keeps changing in these commercial and solvers questions. So there is a way we are not digging into each, but there is a way, for example, to change the parameters. For example, you have a given problem and for some reason you know that first should be used during the algorithm processing for example. So you can include this information when you start your, for example, your problem resolution via gums. For example you can give this information. Please use that first search strategy. So basically you give you build what we call a text file with all the configuration of the branching bound message, everything you want to change in the branching bounds, changing from the default strategy. And the method starts with your best configuration. For example, this is a little bit more sophisticated, but this can be done as well. That's it. Yay! Do you have questions? Many. So you have graphic solution methods enclosed. A little bit more about the simplex a little bit more about branching bounds okay. Guys still here? What a. Graphic solution method. It's very basic. You can read basically everything that we discussed. Simplex is a little bit more advanced because it needs some concepts that I didn't discuss. For example, basic solution. Well, we discussed, but in a very quick way. And the branching bounds description it's very basic to eventually the terminology will be a bit different because different textbooks they eventually can use different terminology for example. Terminates the node or to prune a node archibong node. But it's the same principle. Okay. My suggestion is. My suggestion is to go. As much as you can on this inclusive material. In terms of final assignment, obviously I'm not I'm not asking you guys to provide you more than I provide here in these slides. Okay? So you have the enclosed material, but it doesn't mean this is for you guys. Learn if you want to learn more. This is not for me. This is for you. Okay? If you don't want to learn anymore, that's fine. But at least the content of these slides should be none. Okay. Should be learned. All right. Those, the number being close to point five. How what sort? It is like. The number being close to point five. It was intended to be used. It's very intuitive because this is the most what's the most distant value from an integer one? Point five. Point to nine is close to 1.1 is close to 0.2 is close to 0.3 is close to 0.4.5. I don't know either zero one and so and so far so exactly the middle of the interval. Very general rule. That rule in our study is whatever is going to be well. So basically this is automatically in general you say like we think the branching bounds. Said we used this particular thing in our question here. I was trying to connect, how did we use that point first thing in our brains and. How you use. So basically we have to analyse. For example, let's assume that you are implementing this algorithm. For example. It's like a rounding algorithm. So you analyse, you have a given value and you have to analyse only the fractional part of a given solution. You analyse if it is closer to five, if, for example, you start branching from it or, or you have a number of fractional parts, you have a list of fractional values. I don't know if this is your question. And you, you can choose from the variables or the variables whose fractional part is close to 0.5. But inside the message is automatic. I mean, it can be in any in any sort of step of the algorithm. Okay. It can be even like the reading or luxation imposing further constraints on the fractional parts of the solutions and things like that. No, no more questions. Okay. That's amazing. Okay, great. Yay! Yay, yay! Thank you very. Much. Have fun. You have the whole week. This one. Can we talk about our talking? Can we talk about our topic? Yeah. Very quickly, because I'm tired. You talk. But where is your group? Yeah. Okay. Can I use the. Yes. Also. It's like a wedding and we have a lot of guards, so we try to start again. You have what. A wedding, okay a wedding. So we try to divide different people into different groups so they sit on the same table. That's like one table for maximum rate, like a person. And then we try to thank you. They.