f20_report.md

Fall 2020 Lab Survey

Numerical Questions

	question	NA	1	2	3	4	5	avg
0	None	None	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	None
1	Lab significantly reinforces the material taught in the lectures.	1	0	1	5	36	34	4.36
2	Through lab, I learned how to apply knowledge from this course to real-world problems.\n	1	0	1	4	42	29	4.3
3	Lab introduced new content that was not presented in the lectures.	1	0	13	20	26	17	3.62
4	I looked forward to attending lab each week.	1	4	10	20	34	8	3.42
5	Prior to this class, I had no experience with programming.	1	29	27	6	5	9	2.18
6	Prior to this class, I had no experience with Jupyter Notebooks.	1	7	8	2	10	49	4.13
7	The Jupyter Notebook labs were appropriate for my programming background.	1	1	8	10	29	28	3.99
8	The software required for lab was reasonable and requirements were well communicated.	1	2	3	8	32	31	4.14
9	Web-based labs (Traveling Salesman Problem, Shortest Path, and Minimum Spanning Tree) were self-explanatory.	1	0	1	14	25	36	4.26
10	Jupyter Notebook labs were self-explanatory.	1	0	7	25	31	13	3.66
11	I could implement an idea (LP, algorithm, etc.) in a Jupyter Notebook independently.	1	6	14	16	26	14	3.37
12	The python package, gilp, improved my understanding of the simplex algorithm.	2	1	4	5	28	37	4.28
13	The python package, gilp, improved my understanding of the branch and bound algorithm.	3	1	4	10	30	29	4.11
14	Understanding the geometrical interpretation of a linear program helped me better reason about linear programs in general.	2	0	1	7	29	38	4.39
15	Understanding the geometrical interpretation of the simplex and branch bound algorithms helped me understand the mechanics of these algorithms.	2	1	1	10	34	29	4.19
16	Prior to seeing visualizations produced by gilp, I had a strong understanding of the geometrical interpretation of linear programs, simplex, and branch bound.	2	2	11	34	19	9	3.29
17	The python package, gilp, taught me the geometrical interpretation of LPs, simplex, and branch bound.	2	0	2	13	42	18	4.01

	question	NA	1	2	3	4	5	avg
0	None	None	Too Hard	None	Just Right	None	Too Easy	NaN
1	Did labs challenge you to an appropriate level?	1	1	25	45	4	1	2.72

	question	NA	1	2	3	4	5	avg
0	None	None	Easy	None	None	None	Hard	NaN
1	On a scale of 1 to 5, how challenging was installing the software required for labs?	1	18	26	16	13	3	2.43

	question	NA	1	2	3	4	5	avg
0	None	None	Poor	None	None	None	Clear	NaN
1	On a scale of 1 to 5, how clear were the instructions for software installation?	1	2	6	10	27	31	4.04

	question	Web-based	Jupyter Notebook
19	Which style of lab did you prefer?	21	55

Appearances under "Top 3 Labs"

	0
First Year Writing Seminars	33
Seat Packing	25
Baseball Elimination	24
Minimum Spanning Tree	23
Diet Problem	21
Simplex	19
Shortest Path	17
Maximum Flow	16
LP Formulation	14
Traveling Salesman Problem	13
Transportation	13
Branch & Bound + Knapsack	6

Appearances under "Bottom 3 Labs"

	0
Branch & Bound + Knapsack	32
Traveling Salesman Problem	25
Seat Packing	23
Transportation	21
Baseball Elimination	18
Simplex	18
LP Formulation	18
First Year Writing Seminars	13

	0
Shortest Path	11
Maximum Flow	11
Diet Problem	10
Minimum Spanning Tree	8

Text Questions

Why were those three labs your favorite?

The labs covered my three favorite topics from the course.

They related the concepts we were learning to concrete problems, and how to adapt them to different problems

Learning how to apply the methods I had learned in class in programming was interesting.

I enjoyed the transportation lab and the seat packing lab because they seemed most applicable to the real world. The LP Formulation lab was most enjoyable for me because it seemed to be the most versatile in its use and there was more to think about compared to the other problems.

I enjoyed the visuals that we saw in the shortest path lab. And the other two I chose because I thought they were super interesting and applied to real life.

I am not really sure, I pretty much enjoyed all labs equally. I suppose the FWS and seat packing were most applicable to the real world surrounding me, so it was the most interesting.

A lot of work went in to make and perfect these labs. They were in depth and quite frankly some of the most fun assignments I've ever done. From the complex numerical solvers to the thorough simulation software, these labs truly aided my understanding of the curriculum and sparked my interest further in OR. The labs truly helped me appreciate the power in computation today and how it's used to solve problems in the real world.

They are useful to our real life.

These labs seemed the most applicable - particularly the writing seminar lab and the seat packing lab because they were specific to Cornell and thus made the labs super applicable.

I liked seeing all the applications of the algorithms!

The diet problem was really cool because it was the most applicable to my life, and I was also pleased with how far I've come with my growth in the course, exhibited through this lab. The other two were fun because they weren't too challenging and I could finish them in a reasonable amount of time.

These were the 3 that were the best at making a real world example and reinforcing info from lecture.

tsp was just fun to watch.

Diet problem was interessting and showed how I can use OR to live better.

Maximum flow had many nice graphics

For some reason, those labs just came very naturally to me. Perhaps it was the way it was written? I'm not too sure. Those labs were very applicable and those labs simply just logically flowed well.

They were short and fun, pretty straight forward.

They seemed to have the most real world applications. Also the simplex one really taught me how to apply the simplex method.

Those labs helped me with understanding the weekly homework assignment for that unit

I just found linear programming to be interesting/fun in general.

They were the most engaging because they had the most real world applications, so I actually understood why this material was useful.

They made the most sense

I thought that they were pretty well-organized and easy to follow in comparison to the other labs.

For the Shortest Path and Minimum Spanning Tree, I enjoyed the use of external software to play around with the algorithms a bit more freely than in Jupyter Notebook (I find it kind of clunky to use in that regard). Jupyter leaves a lot of the processing behind the scenes, where the greater degree of visualization made those labs feel more interactive. As for Branch & Bound/Knapsack, I felt it was one of the smoothest-running notebooks and flowed very well, making the concept easy to follow and understand.

The diet lab was one of the more interesting applications to real-world problems using ORIE. The Simplex and shortest path labs allowed me to better visualize the algorithms and strategies we learned in lecture.

Greatly increased my understanding.

I understood the content the most and therefore was able to get more out of the lab.

I liked the packets we had to fill out, and exploring the algorithms through that method.

Those three labs are more favorite because I found problems interesting and challenging.

Shortest Path - Getting to use the online OR tools was fun.

Simplex - Short and simple

Diet Problem - Got to really apply the model into a real-life situation

The baseball elimination problem is just a fun type of problem in my opinion. For the LP formulation and the seat packing, it was clear how this would be useful in the future so it gave me more motivation to really focus and try to understand them. The seat packing one was especially cool to see how the OR department helped model the lecture halls for covid.

They had very interesting visualizations and demos that I found very stimulating and engaging. It was a lot of fun to play with them while also enriching my educational experience.

I found a lot of the LP labs to be really frustrating and I would always have issues understanding how to properly set the constraints. If I misunderstood something, it tended to be really frustrating because I coudl not move on with the lab.

they were cool

I liked the shortest path lab simply because I like Dijkstra's algorithm since it's very applicable to the real world where GPS would use it to find the shortest path. I liked the LP Formulation lab because I was struggling to understand how LPs worked, but the lab helped solidify my understanding. Lastly, I chose the seat packing lab because I found the visuals interesting (and it felt close to home given the COVID situation).

I really enjoyed these labs because it felt as if these labs caused me the least amount of technical issues when it came to actually doing the work required for lab so it was easier to just focus on the content.

They applied the models that we had been learning about to real problems.

I personally though the seat packing lab and first year writing seminars were interesting because it demonstrated real life applications of the materials we were learning that affected our experience at school. I also thought it was interesting to see how the algorithms could be used to solve inputs containing very large data sets in a very short amount of time. As for the simplex lab, I enjoyed using gilp to see the geometric interpretation of the algorithm.

they reinforced concepts in class that were initially hard to understand and in the case of seat packing showed a real world application of the models we looked at in class.

I like most labs that reinforces me knowledge.

I liked how they were applicable to Cornell (but I also liked how we had a combination or Cornell related and non Cornell related labs). It was very interesting to see.

I thought what we learned in those labs were really beneficial and I could easily see how those algorithms could apply to many situations in my daily life.

I think first three gives very clear instruction and there's code already written for us to interact with.

i just thought they had the most interesting topics. they were still difficult but in the end when i figured most of it out i liked it a lot

Those three labs were my favourite because of their real world applications interested me most.

Baseball was fun because I enjoy sports. LP because I'm a math person and find mathematical equality interesting. Minimum spanning tree had a cool website.

I like when there is code in them and modelling

I am a bit biased in picking my favorite labs based on how well I understood the material going into the lab. I think that these labs did a good job of reinforcing the material from lecturing by breaking it down (ex: max flow lab had us draw out each step of the Ford-Fulkerson algorithm including the residual graphs) in a way that we could see exactly how it was done. This breakdown is particularly useful when we have to replicate these algorithms by hand on homeworks or exams.

These were three of my favorite topics so it makes sense that the labs were also my favorite.

These three labs all include many graphics, which I like a lot.

For the Max Flow lab, I enjoyed it simply because it was my favorite unit of the course (or at least the most memorable). Therefore, it was an interesting lab that reinforced many key concepts from that unit as well as concepts that may arise later in the course. I enjoyed the FWS lab and Diet Problem since they were both very direct applications of the different problems rather than more abstract versions. Obviously, seeing how FWS sections can be found for students using the models we learned in the course was very interesting and relatable. I enjoyed the Diet Problem lab for similar reasons.

These were the problems I found the most appealing to start with, so it makes sense that I liked the labs for them best.

Those labs were my favorite because I liked the labs where we applied our knowledge to something that closely resembled a real-life scenario. In terms of the branch and bound lab, I also really enjoyed seeing the process of branch and bound as applied to a graph and it definitely reinforced my understanding of the material.

They seemed to make the most sense and reinforced the ideas taught in the lectures.

I love the real world applications that were directly shown through these labs.

The web-based ones were more interesting to use and clearer. I think they helped with my understanding better.

I found these the most interesting because I thought these labs have the most application as I can even apply them on different concepts such as the problems we have gone over in class before.

They were satisfying to solve.

I had previously learned the shortest path algorithm and it was interesting to see it applied in this different context. I liked the transportation lab because it involved the Cornell Campus, and I liked the simplex lab because I liked the visualizations provided to see the points on the constraint lines.

I liked the first few because they didn't involve the coding. Doing the algorithms by hand was the best way to learn them fully. I also liked the diet one because it is most recent in memory and was also a really nice application of the algorithm.

These three labs were my favorite since I enjoyed the content the most. I found that they were able to be completed well within the time period given and I learned a lot throughout these labs.

I really liked seeing how the algorithms and problems we have been learning in class can be applied to real word problems. I also enjoyed incorporating Python into these labs. I feel more familiar with the language and know it will help me as I take CS next semester.

All three of these labs really enriched my understanding of the topics and were easy to follow.

I just love integer programs and LP programs. Also, these labs were more relatable to the real world so it was really fun to apply what we learned.

The maximum flow lab really helped me to understand the topic and reinforce what we learned in the lectures, while the FWS and diet problem labs were directly applicable to my daily life, so it was interesting to see how the concepts we learned could be used for common problems.

I think those labs all gained the most from using computation to apply more of the ideas were learned in class on a bigger scale. Additionally, I felt that these labs were the most hands on the with the programming, so it helped you better understand how the algorithms are working.

I feel like I learned the most during those labs. They had less coding making it easier to learn the content.

I found those three labs to be most relatable to my life. For example, after doing those three labs, I was able to tell if a team from a soccer league would be eliminated or not, how I would be assigned to FWS in the upcoming Spring semester, and how Cornell designed the seating charts for everyone while maintaining social distancing rules.

I just understood them the best

i liked the labs that drew out visual models so I could visualize the iterations of the algorithm

The earlier two showed us visual representations of the algorithms we learned as well as the simplex graphical models were interesting

A good usage of technology that were not too tedious in running too many iterations of a certain algorithm.

They were fun and they were the ones I remembered the most.

From my memory, I just enjoyed them the most and helped reinforce the concepts we have learned to real world problems.

Seeing how to construct graphs is interesting and helped me reinforce my knowledge about them

Had real-world, even Cornell-based applications.

I quite liked the visual components of the shortest path and MST labs, and the baseball elimination lab was an interesting application of max-flow.

What could have been done better in your favorite three labs?

In general, this applies to most labs, but if you could indicate better how we should write the code for some of the problems, it would help a lot. Sometimes, it was hard to figure out the exact method or function call we needed to use.

I would have preferred documentation on the programming language and the modules used, but I am aiming for a computer science major, so I may be an exception.

I feel like more notes on syntax within the code blocks would make it easier to understand what is going on.

I suppose in any of the labs I would rather more working through some things together, sometimes it was hard to figure out some questions or exactly what python wanted.

Maybe a little more clarity on what's desired by the questions and possibly fewer response questions and more coding/trial and error questions.

Ask more questions.

I don't think anything needs to be changed

I would have liked to write more of the code myself.

I don't really have any opinion on this, they were the labs that did everything right in my eyes.

Sometimes putting code into python is more annoying than helpful, because we were never truly taught how to use python.

I would appreciate the opprotunity to code on my own. I know that the point is to learn the concept, but knowing how to use these tools is what truly gives power. Since most important problems are large in size. Learning them in a group setting will facilitate many things.

I think they were excellent! On the whole, the labs tend to be on the longer side, so maybe if they were shorter it would be better.

Not much. They were pretty good.

Not sure. I feel like overall most of the labs were very applicable to real life and were fun to do. They were never too hard and I did not feel like they were a waste of time. Good job!

Maybe encouraged collaboration between students more

I don't know, I had no real issues with them at all. Maybe some of them could have been shorter.

Sometimes they were very tedious and long, so it can feel like busy-work.

Nothing

In all labs, really, certain instructions or questions were quite hard to figure out not because of difficulty, but more because of clarity. I think some of the questions being a bit more thorough in their wording, as well as clearer demarcation of changes we have to make in actual code (ie, the comments) would make the labs flow a little better.

I do not remember the specifics, but the questions could always be made easier to understand. For example, there could be less open-ended questions that leaves room for confusion about what constitutes a suitable answer.

More written instructions.

I think my only problem was with the coding part of the labs.

Nothing that I can think of.

No suggestions

Not much really, but maybe clearer instructions?

Nothing. They were my favorite because I felt like they could not have been better.

The knapsack problem lab seemed to be pretty dense to me and hard to follow, even though I found the concept and content to be interesting.

idk

I think we should have started with Jupyter from the start. I prefer that over the web-based.

I'm not sure how feasible this would be but having optional extra sections for every lab would have been appreciated because they are useful to go back to just to review the information after the lab to get a better understanding of the lab.

-

I think the part in the simplex lab about the pivot rules wasn't very clear in that I didn't really know what to do just from the geometric visualization alone.

not much

I can answer questions more details.

Not sure if this is application to just these three labs, but sometimes I felt like the labs were long and tedious (especially on days we had to go through two different jupyter notebooks in one section).

Some of the coding instructions could be more clear since I found myself asking the TA a lot about that aspect of the labs.

maybe make them a little shorter. also i recommend visuals for every lab they're very helpful

These labs could have been improved if there was greater incentive to do some of the Bonus parts for some of the labs.

Not anything specific for these labs but sometimes the coding sections were difficult to figure out

Maybe less hard coding to write

As someone who has no experience in coding, I often got lost trying to fill in lines of code. I recognize that figuring out what to put is the challenge of the lab, but sometimes I would get really confused with all the variables.

I enjoyed them, I do not think they could have really been done better.

The code modification part of the lab is hard. This isn't suppose to be a course about coding in my opinion. Other than that, the labs are really cool.

From what I remember of them, they were very well executed and, frankly, I have no feedback on how to make these significantly better.

I think a lot of the labs were much longer and drawn out than they needed to be.

It might have been better if we could have tried out some more code because I thought that using code applied to OR problems was a useful skill to get some practice with.

For the ones that used Jupyter Notebook, possibly more help with technical issues and syntax for the code.

I wish there was more of a collaborative nature in all the labs.

I think the lab activities could have been shorter or more relaxed so we could understand the web models more.

More group work.

I think overall the labs would have been better if they were more collaborative. I think we found it difficult to work together in breakout rooms when we were all in the big room together

In my lab there was zero group work actually done. Everyone was paired into a zoom breakout room and no one spoke to their partner or used the chat. This was frustrating because when I had questions my zoom partners would not respond verbally or in the chat. The labs were basically all individual and not fun at all.

Sometimes, the code that was supposed to be added in was out of my range of programming abilities and I required assistance. Overall, the TAs were usually able to help with that.

I think there was some inconsistency in the length of the labs.

Honestly, I believe they were perfect.

I think there could have been more specificity when it came to describing the situation and the problems that we are trying to answer. Sometimes they were just worded in a confusing way.

nothing

Nothing much, I really enjoyed those three.

understand Jupiter notebook more

maybe help us a bit with the python syntax, this was often the most difficult part to figure out

Nothing comes to mind

I am not too sure, I don't remember them in their entirety.

Don't think anything.

I sometimes found it difficult to work collaboratively with COVID and online.

No

Perhaps doing some more of the model-building. I doubt we'll have the OR-tools package from this class in a real problem setting.

I'm not sure that this applies to these particular labs, but some of the labs boiled down to running a cell, recording the output, and making a trivial conclusion based on the data.

Why were those three labs your least favorite? What could have been done better?

The labs felt a little harder to follow along each part

I did not have a particular dislike toward any of the labs since they were all enjoyable and challenging.

The TSP was among my least favorite because it felt the most confusing especially the questions at the end of the packet. The Shortest Path problem seemed a bit too menial labor like and the Maximum Flow lab didn't consist of any clear application use. I think one thing that could be made better is to have a sample answer and demonstrate why a sample answer is a good answer to a question. Some of the steps were also unclear in the web-based programs when pared with the handout.

I did not have any labs that were my least favorite, but I do not think I had understood the material well enough going into the branch and bound lab

I am honestly not sure about this one, I just remember these as being rather long ones (I may be mistaken), and so they were hard to get through in the time given.

These labs were not awful in content, but could use a little cleaning up. Instead of asking 40 short questions, maybe ask 25 more analytical questions. I feel a lot of these threw a lot of data at us, which is fine, but some checkpoints thrown in from time to time would be nice to ensure that we are fully grasping all the aspects of the problem and code.

Sometimes I cannot install the software successfully.

The TSP lab was just longer than the others which made it harder to complete on time. The other two labs (Transportation and Simplex) weren't necessarily bad, but I just found them less interesting compared to all the other labs.

I think, out of necessity, the first few labs were "lighter" in terms of content. I also don't think I understood Branch and Bound very well until I read the handout.

These were really hard to understand. The labs were super long and the explanation was insufficient to motivate proper understanding. I didn't take much away from these labs.

These labs all used gilp, which never really worked for me.

FWS was not very interesting Baseball was a repeat.branch and bound didn'tt work at all

I think these labs were simply too long. I remember slugging through them, and it honestly was a bit stressful to try to finish. I think if they were just shorter, it would be a lot more enjoyable.

The seat packing one was pretty boring. I was really confused on the LP Formulation lab. And the TSP lab was good, but didn't explain how the last heuristic works.

I felt they were very long and somewhat confusing. The TSP one I did not understand all the heuristics yet and they were not all ones we needed to know, or were used in class.

I didn't really understand those, and I had trouble installing ortools. I'm not too familiar with how python works (I've only formally learned Java) and it was really confusing to figure out how to add to the code to compelte the problems

I didn't dislike the lab format, honestly these are just my least favorite topics (which still aren't bad).

Again, it was mainly because these were quite tedious labs.

They were kind of confusing. They could have been explained more or have more hints

I was just pretty confused by the images that were shown in the lab. The TA explanation helped some, but it still took a lot of time for me to understand in comparison to the other labs, which usually go pretty smoothly.

Not that I didn't enjoy Simplex of LP Formulations (much the opposite) but I remember missing part of the lab experience in that the graphical models of the LPs simply didn't load on my laptop, meaning I just had to skip interacting with them and assume the results for the questions.

They're just my least favorite topics, not necessarily a fault of the lab itself.

Slightly complex. Ideas that weren't introduced fully in the lecture.

The first lab was just difficult in terms of figuring out how the labs worked. THe seat packing and branch and bound lab were just a little bit confusing in terms of questions.

I had a hard time with the material and I got confused by the labs rather than having the material reinforced. I wish there had been less coding, and little more of the format of the first three labs.

Maximum flow is my least favorite because I remember struggling with the code for this lab. I would suggest to make some hints for this lab.

Baseball Elimination - Just personally a very confusing concept for me

Transportation - Too long

Branch and Bound - Dull

I think these were my least favorite labs because they were not on Jupiter notebook and they did not have a coding part.

I found the explanations for these labs to be pretty confusing and very dense. It was hard to follow and I found myself asking the TA numerous times for clarification. I also found the code in these labs confusing to debug. Having a better tutorial on the Python packages would result in a better experience with the programming aspect.

I just found that filling in the constraints properly could be really frustrating especially because I would always be missing one thing, but it would mess up the entirety of the lab.

boring and confusing

They were my least favorite because I'm not particularly fond of the topic itself; there's nothing inherently bad about the labs though.

Despite updating packages and getting help and being in my opinion fairly tech savvy, I had to constantly struggle with packages needing to be updated or fixed in order to get the labs working correctly and sometimes even after those fixes the labs would randomly break halfway through.

The simplex lab was very long and I remember finding the first two labs slightly unclear in terms of what was required for answers (though this might have just been because the course was unfamiliar rather than being an innate problem with those labs).

The traveling salesman program and shortest path lab required writing on paper, which was more time-consuming than just typing the answers on jupyter notebook. As for the maximum flow problem, I kind of disliked having the solve the problem by hand, since I already had enough practice with that on the homework. I think just focusing on the computer programming side of things would make the labs more enjoyable.

I didn't really dislike any labs, so i just chose the first 3.

For MacBook user, I can use the Jupiter well in those three labs.

Honestly, these three weren't bad. I thought some of the questions were unclear (I didn't know that they were asking).

The instructions and coding on those were the most confusing.

I think the description of the problem is not easy to understand.

they were just super long i dont know i just didn't like it that much. the seat packing was cool in idea and very important/practical but i think it turned out to be more difficult than i thought. i was going to list the max flow lab becuse it was super long but i also think it was pretty helpful

These three were my least favorite as I enjoyed the Jupyter notebook format much more over paper and website labs.

I struggled with these concepts. Seat packing could have used a bit less code. I feel like a lot of the work was done for me; however, it was very cool to see real applications of the concept.

A lot of code to write and some were just long

Again, I am biased based on my understanding of the material going into the lab. These were the concepts I didn't understand that well, and it was incredibly challenging to figure out the lab without a 100% understanding of the material. Especially for the branch and bound lab, I would get stuck on basic concept (IP is a ___ bound for the LP) without which I couldn't do the lab. Interspersed confirmations of these basics are incredibly useful without giving away all the answers to the labs.

I was stuck with some of the coding for the lab so it took me a little longer even though I knew how to do it logically I could not code it.

The coding part was hard. The questions can be clearer.

These labs were my least favorite mostly because I had the most trouble with these sections in the course. As a result, the labs felt more tedious since I was also having to work through the struggles that I would encounter regarding the subject matter. Some of these labs introduced a few new things that were not explicitly covered in the lecture, which meant I would have to read the course readings during the lab which would interrupt my flow. It is not that I disliked the readings (as they were very comprehensive) or was not planning on reading through them. It was mainly the fact that I struggled with these sections and had to learn the basic abstract version of the problem, while also having to implement it in some real way.

I really never liked the LPs as a whole. probably because I never really understood them properly, so this made these labs incredibly challenging.

Those three labs were my least favorite because I already felt as though I knew some of the material and I didn't feel as though I was branching on my understanding, just repeating what I already knew so it felt quite tedious. Also these were some of the earlier labs with less code, which made them feel longer and more slow.

There were many technical issues, involving me having trouble with the syntax for python.

I did not see the real world application as clearly and they were not as fun to me. I think also the length of these were a little too long.

They were very long and didn't explain how to write the code for Jupiter notebook. There were times when I knew what to do but did not know how to code it using the syntax. I think if we got a mini-lesson in the beginning on some basics for syntax in Jupiter Notebook it would have helped a lot. Some of the labs also introduced new concepts, except they were presented in a way that assumed that it was known.

The instructions were unclear and the code/ website was difficult to use.

I don't really find the TSP very interesting and it felt like we did a lot of it in lecture and then more of it in lab.

long and simplex isn't that exciting.

While these were still fine labs, I just did not enjoy them as much either because they were too long or some of the descriptions/explanations did not make sense. In the most recent diet lab, I got confused on a lot of the questions as the lab did not provide a lot of information. I think more in depth explanations would have helped, especially if the lab introduced topics not directly learned in lecture.

These were just the labs I had a bit more trouble following along in.

I felt that both of these labs took significantly longer than any of the other labs. As someone who has never coded before, I feel they had code that I was not confident in and/ or had never learned.

These labs were my least favorite probably because they were either less code intensive or they were super long. I think maybe if there were more coding sections and less questions to answer.

These were much harder to understand than the other labs as well as much longer, especially the branch and bound + knapsack lab. There could have been better explanations throughout the lab to improve it.

I think there were pretty repetitive and they strayed a lot from what we learned from class, so it was challenging to grasp what we were trying to solve in that short amount of time.

These were my least favorite because the code necessary to complete the lab made it hard to understand what I was missing in my conceptual knowledge.

They were not my least favorite for being the least interesting but for being the longest and very demanding. I would have to type out the equations for each iteration or some sort, and that took a lot of time.

harder to follow

All the labs were pretty good these were just more boring or really simple and I didn't learn a lot.

The software gave troubles for some of these labs

TSP I just was not the best at since it was my first and I didn't understand concepts fully yet, but the other two labs just were very tedious and made you run through many questions doing very similar things to finish out an algorithm.

I just remember being confused. More explanation on the different iterations of simplex.

The first three were my least favorite because I really enjoyed engaging with code and with jupyter notebook.

A bit tedious.

I didn't like the ones that hammered a point for too long. It felt like a slog and had marginal returns after a while.

I just didn't care for the topics much. That's a me problem, the labs were fine on the whole.

How did you feel about the use of Jupyter Notebooks in lab? What did you like/dislike? What could have been done better?

Jupyter Notebooks were interesting since they provided real world applications of the methods we learned in lecture. I liked the visual and programming aspect since the visual aspect made it easy to understand, and the programming aspect appeals to my ambitions in computer science.

I really did like the use of the Jupyter Notebooks and I would like to have been able to interact more with the code in the notebook. Of course, not everyone in the course has programming experience so it is understandable as to why this may not be possible. But I would like to learn more about how these problems get solved. (like what is in ORtools?)

I really liked it. I am very interested in computer science, and I really enjoyed seeing how these algorithms are coded and processed by computers.

I thought it was a useful tool for seeing solutions without having to do every little step ourselves, however, we were throwing into python labs without any training, so I would often have to ask for help with the syntax (despite programming experience in python). Perhaps if you are going to use python in the future, dedicate a lecture or lab session to familiarizing students with Jupyter notebooks and syntax.

Jupyter Notebooks was great. It's a powerful coding firmware and has it's benefits. Loading packages sometimes was difficult and uploading files into the virtual desktop is not the easiest, but it was worth all the minor inconveniences. If there was a way to upload files easily to the notebook and keeping the file directory in tact, that would be nice. If not, at least a message in the beginning of the lab to check directories would be helpful. Also a note saying to load in ENGRI_1101 from terminal for the labs that need ortools would be helpful.

It is a useful tool for OR.

I really liked using Juptyer Notebooks because now, I feel like I can go home and solve real world problems just given our class experience and the software that we were provided.

I liked them a lot! They helped us understand how the algorithms might be implemented in a real-world setting. I think that students with no programming experience could have used a little more guidance though.

I like the fact that we got exposure to Jupyter notebooks. I've heard that they're used a lot especially in data science contexts, so this might be a useful experience as I grow into whatever field I end up choosing to pursue.

Since there was no specific instruction for uaing python, a lot of the time I had to just ask the TA for what to do and not learn.

I think more time could've been dedicated to them. They are a great powerful tool and definetly would have helped if we understood them better. Could've been helpful to teach us that kernel needs to be restarted after installing a module

I think the Jupyter notebooks were great! The only issue I saw was that oftentimes error would occur, and, as someone with little Pyton experience, it made it awfully difficult to navigate. Some of the errors/packages are not necessarily self-explanatory, so I think next time the amount of coding the students have to do should be minimized.

It was good, I liked it. Being able to solve real problems was good.

I liked them. I like how they led me and they flowed very well.

it was ok

I liked the inclusion. Honestly I don't have any complaints about it.

I really enjoyed Jupyter notes books because you could type answers and run code together. It was the perfect way to teach these concepts. Maybe there could've been a coding tutorial for those who weren't familiar.

I liked them and how simple they were, but sometimes the instructions are confusing.

I liked using them; however, this is mostly due to coming from a CS background. Often, because my fellow classmates didn't have programming experience, it was a little hard to pair up with other people at different skill levels. I think having a solution set for the code, for people to consult if they get really really stuck would be helpful.

Knowing this is a new feature of the course as well as the difficult semester its been, I don't blame the notebooks for any technical difficulties, but they were certainly present in many labs. Besides that, I just wish the instruction wording was more thorough and that we had slightly more guidance in the addition of our own code, not that it was impossible to figure out, but I had to guess at many times if "that's how Python works" since I'm more experienced in Java. It'd probably be easier for many people if the code-addition sections were either even more explicit in their instructions, or if we had a little more introduction to Python (which I know isn't the point of the course).

I had never used Jupyter Notebooks, but it turned out to work pretty well. Forgetting to import something or execute a block and having to search back through the lab was annoying, but not necessarily a fault of the lab.

I liked Jupyter Notebooks, I thought it was a great way to get hands-on experience

Jupyter Notebook was able to present all of the material in a very organized manner. I think that it was due to my lack of programming background that caused me to struggle a bit when completing the code questions in lab.

I had difficulty with Jupyter Notebooks because I had no prior coding experience, so it often made me more confused because I didn't know the format when I was supposed to edit the code. I wish there had more instruction on the format of the coding.

I really liked using Jupyter Notebooks in lab. In comparison to web based labs, Jupyter labs require less hand writing and allows more space to experiment with the input.

I liked using Jupyter Notebooks in lab. They were very organized and it was so nice to have everything in one place.

I like Jupiter notebook a lot. Combining coding with the problems we did in class was just a fun way of making it seem more real.

I thought they were really interesting and helped me connect the concepts in lecture with applications in the real world. It was also interesting to see how the solutions are optimized through algorithms. However, they sometimes became dense and convoluted and the programming parts could be better explained in the later labs. Overall, I like the addition of the Jupyter Notebooks to the class.

It would be useful if the documentation was more readily available for the packages we used so that we didnt have to guess about syntax.

they were reallly confusing. I didnt know how to use them and I kept having trouble with errors and import failures almost every other lab

I really liked the use of Jupyter Notebooks. I ran into quite a bit of installation issues, but I have experience with debugging such stuff so I was fine in the end. I'm also a bit biased on that end because I have advanced knowledge in Python.

I stated this somewhat earlier but I feel like dealing with the technical issues inherent to Jupyter notebooks was a bit of a challenge but I am not sure how to entirely fixed it. It was useful in creating an interactive environment though.

The notebooks were fine. I think slightly more intro into the workings and/or structure of the code we made use of to solve models would have been interesting and might have been beneficial.

I liked how the Jupyter Notebooks were easy to submit on Gradescope, and how it efficient it was to type out the answers. I also liked being able to use some basic programming skills and to get firsthand experience with using software to solve the problems on a computer. However, sometimes it was also kind of hard to install the software and occasionally there would be a need to debug the code, which I didn't like that much. I think what could have been improved were the instructions on some of the questions, since I felt they were worded a bit ambiguously at times.

They were easy to follow and helped reinforce the class material since the problems usually involved different contexts.

I like it because given instructions are very easy.

I thought it was a good idea!

I thought being introduced to this type of tool was very helpful. However, some of the explanations of how to add certain code into the functions could have been expressed more clearly.

I think it's pretty good in visualization.

I liked them i thought i had an ok experience. Even thought i hadn't used it before it was good to gain experience in a new software. I'm more familiar with java than python so the spacing was definitely a learning curve for me. There was one lab i had an error and could not figure it out...turns out it was because of my spacing/indentation. Overall jupyter notebooks was pretty straight forward.

I really enjoyed Jupyter Notebooks in lab, they allowed me to grow my understanding of programming and visualize how what I learn in lecture can be used by operations research professionals.

The coding expectations and basics could have been explained more thoroughly at the beginning of the semester.

They were easy to use and practical to use. I disliked installing it.

I liked the Notebooks as an introduction to coding, especially when the code templates were in the lab and we had to fill them in appropriately. The fill-in-the-blank style of lab questions were a great way to apply knowledge without a background in coding.

I liked the fact that we could run code and add to code and it would give us interesting programs when we ran it. I did not like how sometimes it was unclear what the syntax was to code the programs so even though I knew how to solve the problem I sometimes did not know how to write it.

I don't like the use of Jupyter Notebooks because often I was stuck on how to modify the code, yet this should not be a coding course. The lab should not ask students too many coding question.

I enjoyed the Jupyter Notebooks since they were an easy way to compute different programs for problems while also allowing for an easy way to answer questions. Since I had virtually no experience with writing code, one (very) minor problem was that there were a few syntax things that I struggled to get on the first try. Naturally, they were simple syntax errors on my part and easily fixed, but this was basically the only gripe I had with Jupyter Notebooks.

I always had errors with these, it was not a fun time. I think the way to program them without errors needed to be laid out more clearly

I really liked using Jupyter Notebooks in lab, I think it could have been helpful to have one lab early on to explain some of the basic code we would need (e.g m.Minimize(), m.Add(),etc), so that we could have some background and then go on to do more interesting/slightly more advanced coding in future labs.

I liked that you could code certain things, but there were so many times technical issues, or you would run into a syntax problem that you just wouldn't be able to solve, which was super frustrating sometimes.

I did not have any coding experience really outside of Java so I thought this was pretty hard and downloading the software was confusing to me. I think in the future there should be more time dedicated to figuring out the software before the labs.

I think Jupiter Notebook helped and since it solved some things automatically, we didn't have to use time just solving things and we could move on to deeper understanding. Since it was new, there were times when I knew what to do content-wise but did not know how to code it using the syntax. I think if we got a mini-lesson in the beginning on some basics for syntax in Jupiter Notebook it would have helped a lot.

It was very useful - being able to see the results of the coding increased my understanding of the various labs.

I thought they were easy to download and use. There was some difficulty in importing certain programs but for the most part these difficulties/errors were easily resolved

I had little programming experience and no python experience. I did not understand the syntax and was the coding parts I only solved with help from TAs and by recognizing patterns. A lesson on the syntax would be very helpful. I also feel I could've learned some coding if it was introduced properly. Instead, the parts where I had to write some lines were stressful and I felt ill-equipped.

Other than the parts when I had to code, it was very helpful to see the coding implementation and the visuals.

Overall I think the use of Jupyter Notebooks was really helpful and made the labs much more interactive than the web-based/paper labs. I think either more explanation of what type of code is expected would be helpful, especially for those with no Python experience.

I am really glad I gained experience using Python and Jupyter Notebooks and how it can be used in OR. I think additional explanations and background on the program and on Python syntax would help future students.

I feel that students who had experience with coding/ Jupyter Notebook had a significant advantage over students who didn't. Perhaps pairing students with experience with students without experience could have been a way to negate this issue, but I understand this is difficult since half the students were remote. I definitely feel I learned a lot about python that will help when I take CS 1110 in the spring, but really struggled writing code without a lot of guidance or any prior experience.

Loved it. Maybe emphasize a bit more on Jupyter Notebook. You get a sense of excitement when you're able to write out an LP by yourself on Jupyter Notebook (I think being able to write an LP yourself on Jupyter Notebook is pretty important). I think what could be done better would be to maybe add lectures/labs on writing python in Jupyter Notebook.

I like the use of Jupyter Notebooks in lab because they helped me visualize what the code did so that the concepts could be further cemented. Also, the interface was easy to use. However, I did have some trouble with installation/an update in the middle of the semester since I have a windows laptop, so clearer instructions could have been given for that.

I think that it was nice to get to use more computation in our work with Jupyter Notebook. But there were some annoyances when using it. I would be really annoying to run the code if you changed something earlier since all the code would have to run again. I think the instructions on how to implement the code in the lab could be more clear, as well.

I liked that it was all on my computer but did not like the coding aspect.

I liked what I write as a code would run and display a seating chart, branch-and-bound diagram, etc. Sometimes it was challenging to write a correct code to implement, but TAs were helpful whenever I was stuck.

I like it but it was hard

I really liked them. Super easy to vizualize everything and see what was happening. I think you should add a little bit more about specific Python syntax in the labs; We often had questions not about the content, but about the syntax of the code to write.

I like Jupyter Notebook, however some general context in python might have helped

I think it is good to get an introduction into coding, sometimes unclear what needs to be done though which could mess up the purpose of the lab.

I liked them for most part. Annoying to get used to, but once I got the hang it wasn't bad at all.

I really enjoyed using the Juptyer Notebooks in lab. I enjoy coding so I liked seeing how the models we talked about in class could be encoded/programmed.

I like the use of jupyter notebook. It gives me

I liked using them because it's relevant to other classes and it saved paper from the web-based labs. Sometimes inserting images interrupted the workflow but that's about it.

Jupyter Notebooks are great, I had no complaints with them.

Was there any misconception you had about linear programs, simplex, or branch bound that your interaction with gilp visualizations corrected? If yes, please briefly describe the misconception that was addressed.

No

No I think that my idea of what was happening geometrically and what I saw were pretty similar.

I suppose I could never picture 3d LPs before using gilp.

I never truly grasped from lecture that a limiting constraint maximizes a slack variable and that slack variable can then be graphed on a pair of axis (assuming 3 initial variables), which then hits a bound on a feasible region. Seeing the visualization and the iteration and objective value sliders and the planes graphed at every step was very helpful in my understanding of how linear programs look and how the Simplex algorithm truly works. The 3d graphing and geometries helped. This is also the case with the branch and bound 3D models.

No

Not necessarily misconceptions, but the gilp package helped me understand how the algorithms were working (particularly the Simplex method).

The GILP visualizations made me realize that these are very geometric concepts, especially with the cutting plane. I was under the impression it was just a magical algebra box.

As I addressed before, gilp never worked well for me.

I thought LP's were solved the same way everytime, but turnsout there are different ways to go around the edges of a body, gilp made that clear.

I cannot think of a misconception off the top of my head, but the gilp visualizations definitely helped! The only issue was that it took a while for me to install the gilp package. The instructions were relatively unclear, and I remember some of the TA's being stumped as well. I only managed to install it after a student answer on Piazza showed me a different way.

No misconceptions.

No.

i had trouble installing gilp so the TA told me to use the older version because one of the labs had an older version available but after that I wasn't sure

no

I thought that the visualizations in lecture were sufficient for my understanding. It was fun to do it from a coding perspective, though.

-

For some reason I originally thought that adding additional constraints for branch and bound made us solve an entirely new program, but I saw through gilp that it was just cutting the existing feasible region into different portions.

My understanding of the way in which the feasible region was split by constraints in branch & bound was strengthened.

There were no major misconceptions, but it was nice to see everything visually since I am more of a visual learner.

No, they just helped me understand the solutions better.

No

Yes. I generally had a hard time visualizing 3-dimensional planes. GLIP helped me a lot because it visualized for me.

No.

No

No, there were not.

No

-

No, I didn't have any major misconceptions before seeing the gilp visualizations.

no

No.

no

when we were talking about the cut planes that was a little hard to understand and seeing the models/ visuals helped me to better understand it

Yes, the visualizations of the Branch and Bound nodes helped me understanding all the reasons for why a node can be fathomed, not just that it is an Integer Solution.

The 3D models were the ones that I was the most impressed by. Previously I had no concept of the higher dimensional visualizations for theme (>2D).

no

I really liked the graphs, they made it very clear exactly what was happening when we were solving these types of problems from a logical perspective.

No, there was not any misconception of this type.

None in particular.

Not really, I feel like I still dont fully understand linear programs

I don't know if I had any specific misconceptions but I feel as though I didn't have much of a conception of graphical interpretations to begin with before exploring it in lab.

I really liked going through it and actually seeing the processes. In class it can get a little confusing so coupling class with the lab is very beneficial.

None

No

I don't think I had any particular misconceptions.

nah

I was confused about the branch and bound cut constraints, but after seeing the gilp visualization I understood how it got rid of some linear feasible points.

Which nodes to explore for branch and bound.

Overall, I just feel that gilp helped with visualizing the LP iterations as they run. Before, I didn't really understand how the iterations looked on a graph, but running these through gilp and seeing the different paths I could take to get to the optimal solution really helped my understanding.

Nope, there were no misconceptions. The graphical representations were super helpful. Also, maybe some of the branch and bound gilp visualizations were a bit confusing (maybe reorganize them and give more context for each branch and bound geometric visualization).

No misconceptions

I didn't properly understand how the iso profit lines worked before so using the slider with gilp helped me visualize how they work and shifted.

no

No, I did not have any misconception about those topics.

not really just reinforced understanding

Not really, but it was good to strengthen my understanding through visualization

no

Yes it more easily showed how the simplex programs were solved going from each constraint to the next and moving along the line to solve the optimal solution.

Yes, it was great to see the "snake" traveling across the different axis.

No

No

None that I can think of.

What did you particularly enjoy about the gilp components of the lab?

I enjoyed moving the sliders and seeing the methods work visually.

The showcasing step-by-step of what was going on for each iteration was definitely very helpful and very interesting.

I am not the most visual person so it was really nice to see it

I enjoyed seeing graphs and how they changed throughout the process, especially in branch and bound when it was split into regions.

Seeing the feasible regions, planar constraints, and planar objective function of an LP was fascinating and aided to my understanding of the course. Seeing the iterations of branch and bound play out also aided.

It is a useful visualization tool.

I enjoyed that gilp was very easy to work with and it's easy to understand how to manipulate and use the package.

I really liked how easily we could manipulate the visualizations.

They were very intuitive, and the methods were self-explanatory.

Nothing

zooming in and out of the 3d shape was fun

I think the gilp components are a wonderful visualization. I really had a lot of fun playing with it!

The gilp components helped me visualize branch and bound, specifically.

I liked the visuals

not sure

Helped with visualizations/intuitive understanding.

The fact that I could see the algebra I was writing come to life as a real image, and I could visualize my solution to make sure that I was right. This gave me a lot of peace of mind.

The visual features made it easier to understand

It certainly helped clarify just what simplex, branch and bound, etc. were "doing" in the modification of the LP.

It was one of the more interesting uses of code, as opposed to doing m.solve() and looking at a couple of rows of numbers.

The multidimensional representations of the LP feasible regions.

Being able to see everything visually allowed me to better understand the concepts, especially the LP and simplex.

The ability to be able to go through different iterations and see how the feasible solutions progressed.

That I could resize, rotate, and interact with 3d visualizations

The visualization and how I can interact with it freely.

The visuals were clear and made it so that I didn't have to spend the time drawing them out on paper.

I am a visual learner when it comes to connecting algebraic concepts to their geometric counterparts so it was very helpful to not only have visual cues, but interactive ones at that. I found gilp to be very useful in my understanding.

I liked the fact that they were a bit more interactive with how you could see the progression of the algorithm.

I liked the visualization that were provided since it allowed us to get a second frame of reference for some of the things we were doing.

It was an extremely useful tool for understanding how these 'abstract' models that we had been dealing with actually translated to a more 'tangible' geometric model.

I liked how you could adjust the iteration number for the simplex algorithm and see how the algorithm progressed by increasing along each edge.

the visuals aided in understanding class material and were overall very clear what was happening

I enjoy everything.

I liked seeing the simplex visualizations.

I liked that they were interactive and allowed us to see how changing certain constraints and factors affected the optimal solution.

we can play around to find optimal objective value

I just liked to see the visual to what we were doing because it made it feel like i was actually coding something functional

The gilp components of the lab helped me visualize the simplex constraints for a linear program and this was useful and visualizing the feasible area for 3-var problems.

Visualizing the plotting component

They were nice for understanding

It is useful to see graphical representation of the linear programs because you can see the feasible region, variables, and constraints and how they affect the optimal solution.

I enjoyed how it added a visualization of what we were doing.

I enjoyed how I can see the program helping me to find the optimal solution for some problems.

They were a very visual way of keeping track of a large amount of variables and constraints. This helped me to see what was happening when a constraint was met or not met and helped me to extend that idea to problems with more variables that could not necessarily be graphed this way.

Being able to visualize certain concepts.

I thought the branch and bound representation was really interesting because it really helped solidify my understanding and made me understand the potential applications of IP.

Being able to visualize

I like the visual layout. It helped improve my understanding.

It was cool to see the visual.

The visualization made it easier to understand how the different parts worked together.

I liked the slider features, I think it helped me see what was really going on myself.

i liked the visuals

I enjoyed seeing the visual representation since it put it all together for me in my head. Reading the description made sense, but actually seeing it made it click.

How you could move around the geometric interpretations of the linear program and see the model from a variety of angles.

The way it showed each dictionary as you moved through iterations

I enjoyed looking at the graphs since it made it easier for me (a visual learner) to understand LP formulations and constraints.

Visualizing the concepts made it much easier to understand the topics.

It was easy to use and clear when the isoprofit line hit the optimal solution. Additionally, it helped it IP problems as well.

the visual aspect was a nice component

Just the fact that I can see some kind of visual and the fact that I can interact with it, such as sliding the bar when finding the optimal value of a function.

they were fine

being able to see what was happening in the algorithm geometrically

It helps visualize what the things we learn in lecture come out to

They were cool to show how each set of constraints formulated a graph.

Nothing in particular.

I enjoyed the interactivity and comprehensible 3d models that are often hard to draw in notes. Helps to have a 3d model to see the geometric intepretation of simplex, etc. rather than just my drawings in my notes.

Nice to see the visualize version of algorithms.

Particularly helped visualize the different ways you could choose the next constraint to limit in simplex.

Visualizations of data are good points to ask "do I really understand what's going on here or am I just writing numbers and variables down for no reason?"

What did you particularly dislike about the gilp components of the lab?

I did not particularly dislike anything about the gilp components of the lab.

Some gilp components don't work in safari.

I did not like that the graphs did not change for the slack variables, it stayed the same with the first two variables, and we couldn't see how the other variables affected the visual problem.

A little more data from the graphs produced from the package would be nice. For example, hovering over any point and seeing values, not just corner point of the feasible range would be really helpful.

no

I didn't dislike anything, but I would appreciate more comments in the code - this would help me understand what a certain line of the code was doing (even if it wasn't necessary to understand, it would still be helpful).

They wouldn't render inside my Jupyter notebook, so I had to open up a terminal and re-run all the code any time I wanted to see the visualization. And sometimes, it didn't even work.

Most things.

Didn't work in branch and bound

Besides the unclear instructions to install gilp, I don't think I dislike anything! As a visual learner, the gilp components were some of my most favorite portions of the lab.

Nothing really.

Nothing.

importing gilp

Nothing much.

Nothing

Gilp certainly gave me trouble for a few weeks before we updated it a couple of times, and I never got to use the visualization portions to their full extent because it wouldn't render for me (MacBook Pro). I think it was useful for explanation, though.

Especially for the 3D visuals, they were difficult to read and maneuver.

I liked the gilp components.

The branch and bound section was a bit confusing for me.

Nothing really.

-

nothing in particular

The 3D ones sometimes glitched.

None

It was hard to import. My computer kept giving errors when I tried to import it using the exact command that was provided

I'm not sure why but uploading the files for those labs onto Gradescope would usually end up taking longer than normal.

It felt somewhat troublesome to work with at times.

-

Updating gilp seemed to be pretty troublesome at one point, but that seemed to be some sort of software error. Also, when scrolling up or down on the page, there were times where it would instead make me zoom in or zoom out on the 3d graph.

not much

Nothing.

The branch and bound visualization didn't make much sense to me... Also, for the simplex visualization, I think the 3D visualization wasn't working for me? I'm not sure but nothing showed up in the box for me.

Nothing much

nothing off the top of my head

The gilp component part where I had to determine the number of ways I could reach the optimal solution by maximizing a certain variable first was at first difficult to understand even with the visualization.

The 3D models were a bit confusing

nothing

Nothing.

I dislike how I need to modify the code for it to work.

Nothing in particular. These components were greatly helpful for my understanding.

They would often give me errors which were hard to fix

I can't think of anything I particularly disliked.

The technical issues and having to install it.

I think sometimes the coding could get confusing because I did not have much experience before this.

The graphs were kind of confusing sometimes. I think it confused me more than it helped for some questions.

The 3d graphs/formations were difficult to maneuver

I don't think I really disliked anything about them

I didn't like the syntax or coding part but the visuals were nice.

Sometimes the 3D versions of the gilp component made me more confused since I didn't know how to interpret the graph.

I loved gilp components a lot, but I would've liked to learn how to code using gilp.

Sometimes, with gilp the isoprofit lines or the plane didn't move across the graph, so it was hard to learn from at times.

I thought it was difficult to understand what they were saying

None.

sometimes kind of laggy

none

A bit hard to understand the code behind it

Sometimes they were clunky and hard to get specific answers from.

Nothing

No

Some small bugs in looking at graphs of LPs. The slider seemed to reset too easily based on minimal interactions with the rest of the interface and that could make you lose your progress.

Nothing I can think of.

How could gilp be improved to better accommodate students?

I am not sure.

If gilp could work with safari that would be nice because switching to chrome when a Jupyter notebook is already running can be confusing at first.

I am not sure, i don't know enough about gilp

Something that would be really nice is a tutorial over how to work with the software. It was pretty late when I realized that I could rotate the 3D models and play around with some of the features. I feel that if more people knew how to operate with the package, not only would that spark more engagement, but it would assist with understanding the ideas taught in the course.

Have more clear instruction.

I don't think there are any necessary improvements, but it would help if there were more comments in the code.

I had no issues with gilp!

The version issue could be rectified; it looks like the default installed version is incorrect for the class.

Make it work please.

better instructions and better code documentation

Again, the only comment I have on gilp is to clear up the instructions for downloading it. Right now, they are relatively unclear and it was a pain to install it, but I'm sure that, if the instructions are more clear, then every student would thoroughly enjoy playing with the visualizations.

I'm not sure. It was a bit slow on my computer, that's all I can say.

I think it was pretty good.

import gilp instructions

Easier to install/update.

Nothing

Just bit more stability/reliability. I think it was useful overall.

Making it easier to read off corner points and having a zoom slider would be helpful.

No improvements.

I think that it is good the way that it is.

I thought the section where constraints are placed could be a little more clear in terms of which constraint was which.

No suggestions

Better instructions in installing. I had a hard time getting it running on my notebook.

I think it works well now.

I think it was implemented very well and I don't have anything that can improve it

DONT MAKE IT SO COMPICATED TO IMPORT

I think gilp is sufficient as is.

Better documentation regarding what should be done, perhaps like those given in some introductory cs classes.

-

Sometimes the branch and bound labs skipped multiple iterations for the sake of saving time, but that seemed to make the visualizations more confusing rather than helpful. Also, the zooming out feature is kind of unnecessary.

nothing

could be more useful for MacBook users.

Generally it's good.

Gilp could be improved by spending a little more time discussing in class how the gilp works, as long as this is not too difficult to comprehend, it could allow us to better understand its function and importance.

Maybe an explanation beforhand.

not sure

I like the way gilp is right now.

Perhaps a slight amount of control over different things (constraints, coefficient values, steps in optimum/iteration values). This would allow students to vary their input and algorithms without having to learn and rewrite a large amount of code.

Give more of the code so it doesnt break immediately

I think it was explained well when we were first exposed to it (e.g the interpretation of corner points, how to use the slider, etc), so as long as that explanation remains, I think it worked well.

I think it would be helpful to walk through it before starting the labs using it.

I think the use of the visualizations and gilp could be better introduced/explained just so we get more familiar and comfortable using it as a tool to help.

Make 3d models easier to manipulate

I'm not sure, I think it was pretty good

perhaps a lesson about the syntax and coding or how to use it on our own.

For confusing gilp components, like the 3D versions, more explanation about how to interpret the graph could be given.

I think maybe teaching students how to write code with gilp would not only be fun, but it would allow them to understand the module more and maybe even the geometric visualizations for LPs.

I think the animations in 3D can be improved to make it easier for students to see from all angles since sometimes the 3D problems can be really hard to understand.

be more specific about which iterations go with which branch

I think it was a little frustrating when we had to update gilp, but it would not work. Again, TAs were very helpful at this stage.

sorry but i have no clue

its good

Perhaps earlier introduction in labs

I think not much could be done. Pretty good overall.

Not sure

As a student who intends to major in CS, I would love to learn more about how to use OR tools on jupyter notebook

Perhaps showing in more detail the tree traversal of branch and bound. That or provide markers for order of traversal so far so it's easier to keep in your mind.

Nothing I can think of.