Smart Rail Network Optimization Phase 1

by Aidan Macaluso, Max Rothfeder, Valentin de la Pena, Yetayal Tizale

IEMS 313

May 10th, 2023

Table of contents

1 Problem Description

Our team is working with SmartRail, a transportation company that moves shipping containers within its existing network of train stations and tracks. With various shipments having a series of different origin and destination points, we endeavor to help SmartRail create a transportation plan to deliver these shipments at minimal cost. These costs are divided into two components – mileage and reloading costs – with each container having a fixed cost per mile and a fixed cost to be reloaded at each station. There are also constraints within the SmartRail network. Each track can only handle a fixed number of containers moving in both directions combined on a track. Additionally, each reloading zone has a fixed number of containers it can handle - with a station having the possibility of containing multiple reloading zones.

In order to optimize this process, we are creating a MILP (mixed-integer linear program) to model and minimize SmartRail's cost structure in AMPL. We begin the analysis by first analyzing a subset of SmartRail's network – to ensure the MILP is functioning as intended. Then, we will extend this model to SmartRail's current network – containing more shipments, stations, and tracks. To best accommodate SmartRail's needs, the model will enable SmartRail to directly load .csv files into the model in Task 2. In addition to a neat internal output within AMPL and instructions for the client to use the software in the future, we will also provide a comprehensive recommendation to SmartRail based on the data they provided. This recommendation will not only include the recommended paths to transport the containers by rail but also an analysis of the results from the optimization model.

2 Model

As mentioned in the Problem Description, this project seeks to find an optimal solution using a custom MILP implemented in AMPL. This section discusses the components of the MILP and then presents the program itself.

2.1 Overview

This optimization problem will be modeled as a MILP, due to the types of decision variables, objective, and constraints. In order to make this model functional as SmartRail changes and grows their network, we model this MILP data-independently. This is done through the creation of sets and parameters described in sections 2.2 and 2.3 below. Then, decision variables are created in 2.4, enabling the entire model to be written in section 2.6. We explain the derivation of the model more in sections 2.5, 2.7, and 2.8. To actually use this model and compute optimal shipping routes and objective values, the specific data is loaded into the data-independent model. In Task 1, the specific parameter and set values are loaded to the model through a data file (.dat in AMPL). For the client usage in Task 2, the specific sets and parameters will be loaded in as .csv. That will be done in section 3 with instructions in section 4.

2.2 Sets

The first step in the creation of a data-independent model is through the creation of sets. A set is used to describe a component of the model that helps define the dimensions of the problem – within the decision variables and the constraints. For this problem, three sets are needed – a set for each of the tracks (origin, destination), a set for each of the stations (number), and a set for each of the shipments (letter). They are shown in Table 1 below.

Set	Description	Example
stations	The given stations	stations $= \{1, 2, 3, \ldots.\}$
tracks	Listed by origin and destination station	tracks $= \{(1, 3), (3, 1), \ldots.\}$
shipments (abbr. ship)	A list of the shipments	ship $= \{A, B, \ldots.\}$

Table 1. Table of Sets

2.3 Parameters

Next, parameters are needed to define the objective and the constraints (they are given data). For both Task 1 and Task 2, a parameter was created for each component of the .csv files that do not correspond to one of the sets in Table 1 above. This model includes both parameters that are constant and parameters that depend on the sets defined above. Table 2 below defines each parameter used in the model as well as what it depends on (if anything).

Parameter	Description	Units
cost_shipments	Cost to ship.	$ per container per mile
capa_track	Fixed capacity per track.	containers per track
capa_reload	Fixed reloading capacity for all stations.	containers per station
cost_reload	Fixed reloading cost for all stations.	$ per container
orig{shipments}	Origin station by shipment.	(station)
dest{shipments}	Destination station by shipment.	(station)
volume{shipments}	Volume by shipment.	# containers
num_reload{stations}	Number of reloading zones per station.	# reloading zones
x_coord{stations}	X coordinates indexed by station.	(latitude, longitude)
y_coord{stations}	Y coordinates indexed by station.	(latitude, longitude)

Table 2. Table of Parameters

2.4 Decision Variables

In order to find an optimal solution, variables that define the objective function and the constraint are needed. The key for decision variables is that they not only are capable of defining the objective and constraints but also that they can adequately answer the question at hand. The decision variable is shown in Table 3 below. We defined this variable as the number of containers belonging to shipment s traveling along the track from station i to station j. In addition to defining the objective and constraints (see equations (1)-(5) in section 2.6), this decision variable encodes the answer to the question at hand of determining how SmartRail should ship the containers. Combining all the decision variables for a given shipment will detail the path that the containers will follow from their origin to the destination (and can handle a shipment having containers going in multiple directions).

Decision Variable	Description	Example	Explanation
$x_{s,(i,j)}$	Number of containers belonging to shipment s traveling along the track from station i to j.	$x_{A,(1,3)}$	The number of containers in shipment A traveling by track from station 1 to 3.

Table 3. Decision Variable Table

2.5 Optimization Model Explanation

Before explaining what the objective function and the constraints are, it is first essential to define what an optimization model does. The purpose of an optimization problem is to try and find an optimal solution for a given problem (assuming such a feasible solution exists and the objective is bounded). In order to solve this mixed-integer linear program, we will use the CPLEX algorithm in AMPL which will iterate until it finds an optimal solution (assuming one exists; there can also be multiple optimal solutions).

Once the decision variables are defined and the data is provided, such an MILP can be crafted. First, an objective function is needed to either maximize or minimize some combination of the decision variables that define the goal of the project. In this case, as described in the Project Description, the goal of this project is to minimize costs. Then, a series of constraints are created that bound the set of feasible solutions based on either requirements or limitations within the problem. This is explained in section 2.8. In the context of this problem, such limits include the number of containers on a track or the number of containers that can be reloaded at a station.

Combining the sets, parameters, and decision variables into a series of objectives and constraints creates the optimization model – in this case a MILP. The complete model is written out in section 2.6 below.

2.6 Model

Equations (1)-(5) below create the entire MILP. Equation (1) is the objective function, equations (2)-(4) are the main constraints, and equation (5) is the sign and variable type constraint. An explanation of the derivation of this LP and its objectives and constrataints is given in sections 2.7 and 2.8 below. Note that the set 'Shipments' is abbrieviated as 'Ship'.

Mixed-Integer Linear Program

$$min \sum_{s \in \text{Ship}} \sum_{(i,j) \in \text{Tracks}} \left(\sqrt{(\text{x\_coord}_j - \text{x\_coord}_i)^2 + (\text{y\_coord}_j - \text{y\_coord}_i)^2} * (\text{cost\_ship}) * x_{s,(i,j)} \right) + \sum_{s \in \text{Ship}} \sum_{(i,k) \in \text{Tracks}} (\text{cost\_reload} * x_{s,(i,k)}) \quad (1)$$

$$s.t \quad \sum_{s \in \text{Ship}} \sum_{(i,k) \in \text{Tracks}} (x_{s,(i,k)}) \leq \text{capa\_reload} * \text{num\_reload}_k \quad \forall\, k \in \text{Stations} \quad (2)$$

$$\sum_{s \in \text{Ship}} (x_{s,(i,j)} + x_{s,(j,i)}) \leq \text{capa\_track} \quad \forall\, i,j \in \text{Tracks} \quad (3)$$

$$\sum_{(i,k) \in \text{Track}} (x_{s,(i,k)}) - \sum_{(k,j) \in \text{Track}} (x_{s,(k,j)}) = \left(-\begin{cases} \text{volume}_s & \text{if orig}_s = k \\ 0 & \text{Otherwise} \end{cases} \right) \quad \forall\, k \in \text{Stations}, \; s \in \text{Shipments} + \begin{cases} \text{volume}_s & \text{if dest}_s = k \\ 0 & \text{Otherwise} \end{cases} \quad (4)$$

$$x_{s,(i,j)} \geq 0, \text{integer} \quad \forall s \in \text{Shipments}, i,j \in \text{Tracks} \quad (5)$$

2.7 Objective Function Explanation

As described in section 2.6, the objective function (equation (1)) is trying to minimize the total costs within the problem – in terms of the sets, parameters, and decision variables. This costs within this problem are split into two different components – shipping costs and reloading costs.

Therefore, we can use our decision variable $x_{s,(i,j)}$ as the number of containers from shipment s that go from any station i into station j (where a track exists).

First, we look at shipping costs. The decision variables already say the number of containers moving from station i to station j for each shipment. Thus, we want to iterate over every shipment and every track – the two summations. To compute the costs, we multiply the number of containers moving along a given track, cost per mile, and length of track to get the cost to move the containers along such a track for each shipment. The distance formula is used to compute the length of track (the sqrt), the cost per mile is a given parameter – capa_reload, and the decision variables $x_{s,(i,j)}$ encode the number of containers moving along the route. This computes the shipping costs.

Next, we look at the reloading costs. A container has a reload cost every time it arrives at a node (including its final destination). Therefore, we can use our decision variable $x_{s,(i,j)}$ as the number of containers from shipment s that go from any station i into station j (where a track exists). In order to compute this for every shipment and every track, we iterate over every shipment and every track (both summations). This gives the number of containers facing a reload cost, but not the cost itself. Thus, we simply multiply the decision variable by the constant parameter for reloading – cost_reload. This computes the reloading costs.

The final step is to add up the shipping costs and reload costs together. Minimizing the sum of the two costs encodes the objective of the project – to minimize costs – into the objective function (1).

2.8 Constraints Explanation

As described in section 2.5, the purpose of the constraints is to represent either requirements that the model must meet, or limits that the model cannot exceed. There are four constraints in this problem (2)-(5). The first three constraints (2)-(4) are called main constraints, as they specifically relate to the requirements or limitations given by the problem. The last constraint (5) is the sign and variable type constraint, which defines the conditions for the decision variables. The three main constraints relate to the station reload limitations (2), the track limitations (3), and the requirement of balanced flow throughout the model for each shipment (4).

The first constraint is the limit on the number of reloads a station can handle (2). As described in the problem statement, each reloading zone can handle a certain number of containers per day. In addition, a station can have numerous reloading zones. For this reason, the reload constraint should hold for every station (hence the for all k in stations at the end of (2)). A container is reloaded anytime it enters into a station. Our decision variables define the number of containers for any shipment going from one station to another. Thus, we can use the following trick. We want to sum the total number of containers going into a station across all shipments. Since this condition needs to hold for EACH station, we define the decision variable as $x_{s,(i,k)}$, where k is an element of the set of stations we are looping through. The use of the k as both the second element of track as well as the element of stations enables us to find the number of containers going INTO a station over all shipments, for each station. Then, we must define what the reload limit actually is. This is done by multiplying two parameters – the constant number of containers one reloading zone can handle and the number of reloading zones for a given station k. The final step is to use the less than or equal condition, to ensure that the number of containers going into a station is below the reload limit for that station.

The second constraint is the limit on the number of containers a single track can handle (3). Since each track has a capacity limit, this constraint is written for each track within the set of tracks (see condition on the right of (3)). The decision variable says the number of containers from shipment s moving along a given track. Thus, the constraint requires the decision variable to be summed across all s, to ensure the track capacity is done for containers moving from all shipments. However, this only encodes a single directional limit on the track. While we write the constraint for each (i, j), we can use a trick and write the decision variable twice – the first time as $x_{s,(i,j)}$ and the second time as $x_{s,(j,i)}$. By summing for every shipment across both of these forms of the decision variable, we encode the bidirectional requirement into this constraint. The last step is to add a less than or equal sign, and place the parameter defining the maximum capacity for a track (capa_track) on the right. We note that in this phase, the client only has the ability to place one track between two stations.

The third constraint is the flow balance constraint (4). This is necessary to ensure that all of the containers reach their destination and move through the tracks in a feasible manner. Each container belongs to a certain shipment, so this constraint needs to be written for each individual shipment (for all s in shipments). Additionally, there must be flow balance at every single node (for all k in stations).

The left side of the equation handles what is going into and out of each node. The decision variable $x_{s,(i,j)}$ says how much of a shipment is going from one station to another. Therefore, we sum up the number of containers for every track going into station k, $x_{s,(i,k)}$ (term 1 in equation (4)) and then subtract the sum of the number of containers for every track leaving station k, $x_{s,(k,j)}$ (term 2 in equation (4)). By using station k within the index of the decision variable, the number of containers going into and out of each node can be tracked (one node at a time, one shipment at a time). The sign convention used here is positive for containers entering and negative for containers leaving. Doing this for each station and shipment enables the balance to be held for all containers for all shipments at all nodes.

The right side of the equation defines what the balance should be. In the middle of a route (i.e. a transitory node), the balance should be zero (the number of containers going into a station for a shipment must equal the number of containers leaving for the same shipment). However, at the originating station – containers leave the station but do not "arrive" from another track. As mentioned above, the sign convention here is negative for containers leaving a station and positive for containers arriving at a station. Therefore, in the event the station k on the current iteration of this constraint is the origination station, the station should have a balance equal to the negative of the volume for the shipment. The same logic is used to handle the destination station. If the station k on the current iteration of this constraint is the destination station, the station has a balance equal to the positive volume of the shipment. In all other cases, the balance is zero. These two if-else statements create the balance requirement for each (equation (4)).

The final constraint is the sign and variable type constraints on the decision variables (5). Firstly, it does not make sense to ship a negative number of containers on a given route, but it does make sense to ship zero in some cases or more than zero in others. Thus, the decision variables have a non-negativity condition. Secondly, the decision variable represents a shipping container. It does not make sense to ship half of a physical shipping container. Thus, there is also a restriction that the decision variables must be integers.

3 Solutions

Our MILP successfully produced optimal solutions to minimize SmartRail's costs in both tasks, which are presented and analyzed in the sections below.

3.1 Task 1

3.1.1 Optimal Solution

For Task 1, we found the following optimal solution.

Minimum Cost = $1,309,489.28				
Shipments [Company(Track)]	Origin	Destination	Paths	Volume
A(4,7)	4	7	4-6-7	20
			4-1-3-5-7	4
B(3,2)	3	2	3-5-2	5

Table 4. Task 1 Solution Table

Shown above in Table 4 are the results for Task 1. The total cost of handling shipments A and B is $1,309,489.28. The paths that the containers took are visualized in Figure 1 below. We note that all of the routing maps use arrows to demonstrate the movement of containers. Each station is labeled in the form S-R, where S is the station number and R is the number of available reloading zones.

3.1.2 Analysis

As can be seen in Figure 1, all five containers of shipment B travels along the same route from station 3 to station 5 to station 2 (3-5-2). This is the shortest route between stations 3 and station 5, with the only other option being to go around the entire loop (then 2-5). Thus, the optimal solution for the movement of the containers for shipment B is the best possible option for those containers (even independent of track limit or reloading constraints). For shipment A, the 24 containers do not all take the same route from their origin which is station 4 to their final destination which is station 7. This is because of the constraint which limits the total number of containers that can pass along a track. As stated in the problem each track can support a maximum of 20 containers each day. As shown in the final solution, 20 containers travel from station 4-6-7 while the other four travel from 4-1-3-5-7. In addition to being visually shorter (see Figure 1), the fact that 20 containers went the 4-6-7 while only 4 went the 4-1-3-5-7 route (with additional capacity available on that route) demonstrates that the 4-6-7 route is the most efficient. However, shipment A has a volume of 24 containers, meaning not all the containers can take this route due to the track limit constraint. As a result, shipment A is separated into a group of 20 containers, and a group of 4 containers which take a different path to the final destination as shown in Figure 1 and Table 4. In a future step, SmartRail could consider expanding the track capacity between 4-6 and 6-7 to possibly achieve a better outcome (that would depend on the costs for expanding track). Additionally, we note that all reloading constraints were non–binding in this version of the problem.

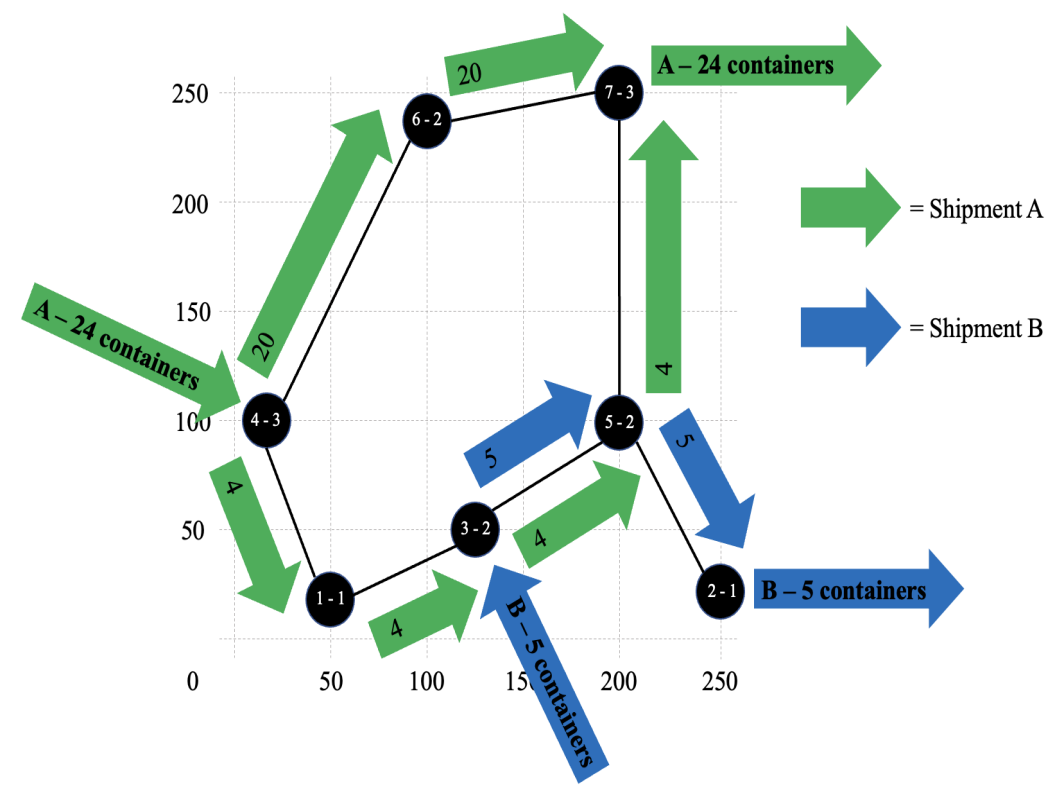


Figure 1. Network Flow for Shipments A (from 4-7) and B(from 3-2)

3.2 Task 2

3.2.1 Optimal Solution

For Task 2, we found the following optimal solution.

Minimum Cost = $3,706,028.92				
Shipments [Company(Track)]	Origin	Destination	Paths	Volume
A(3,2)	3	2	3-14-2	5
A(4,7)	4	7	4-1-3-7	5
			4-6-7	17
B(1,11)	1	11	1-4-8-9-10-11	8
C(13,1)	13	1	13-7-3-1	4
			13-14-3-1	3
C(13,7)	13	7	13-7	2
C(13,8)	13	8	13-14-15-12-4-8	1
D(5,10)	5	10	5-14-15-6-9-10	1
			5-14-15-12-4-8-9-10	1
D(5,6)	5	6	5-7-6	3
			5-14-15-6	3
E(3,13)	3	13	3-7-13	4
E(3,15)	3	15	3-14-15	10

Table 5. Task 2 Solution Table

The results for the optimal SmartRail network are shown in Table 5 above, as well as individual routing maps which can be found in Figures 2-11 below for each of the different shipments. This optimization problem gave a feasible optimal solution for the movement of these ten shipments. The minimum cost for moving them is $3,706,028.92.

3.2.2 Analysis

We will now go through each of the ten shipments, analyzing the way the containers were moved. We note that 6 of the 10 shipments had all containers move along a single path, while 4 of the 10 had containers that split into multiple paths to their final destination. It is also noted that the only costs are for distance traveled and number of stations passed through. Therefore, this optimal solution intends to go through as few stations as possible in the shortest distance possible, subject to the available tracks and constraints. We will follow the individual analysis with an overall discussion of the network flow and possible places to adjust in the future, should SmartRail be able to make such investments. We note that all of the routing maps use arrows to demonstrate the movement of containers. Each station is labeled in the form S-R, where S is the station number and R is the number of available reloading zones.

For the shipment belonging to Company A from station 3 to station 2, all 5 of the containers moved along the path 3-14-2. As can be seen in Figure 2 below, this is the most efficient route for these containers to move, as it is the shortest and goes through the fewest stations.

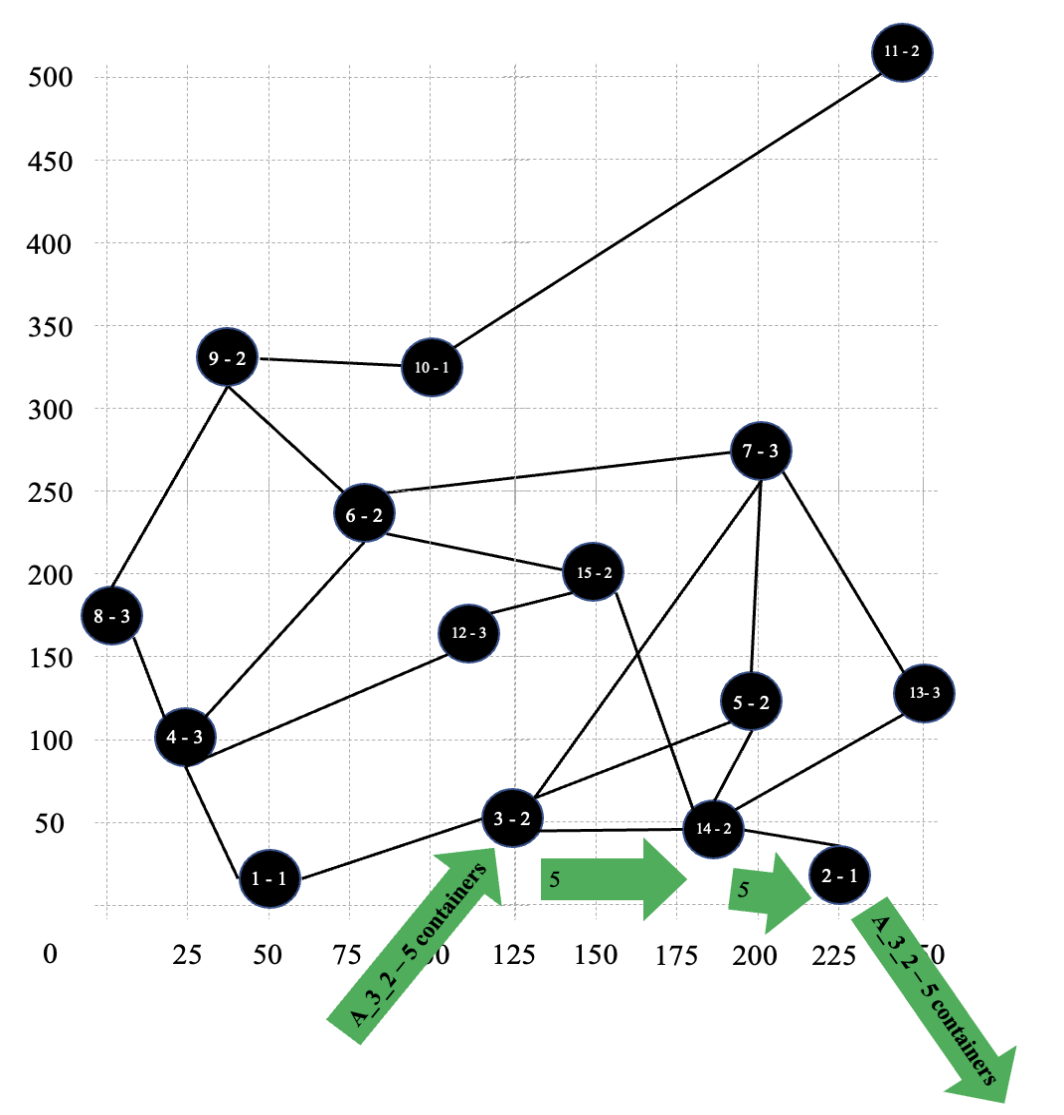


Figure 2. Network Flow for Company A_3_2

For Company A's shipment from station 4 to station 7, the 22 containers are split into two different routes (see Figure 3). The first 17 containers take the most direct route of stations 4-6-3 (see Table 5). The remaining 5 containers take a longer route from stations 4-1-3-7 (see Table 5). The reason for this alternate routing has to do with the reload zone constraints (for station 6) and track limit constraints (for track 6-7).

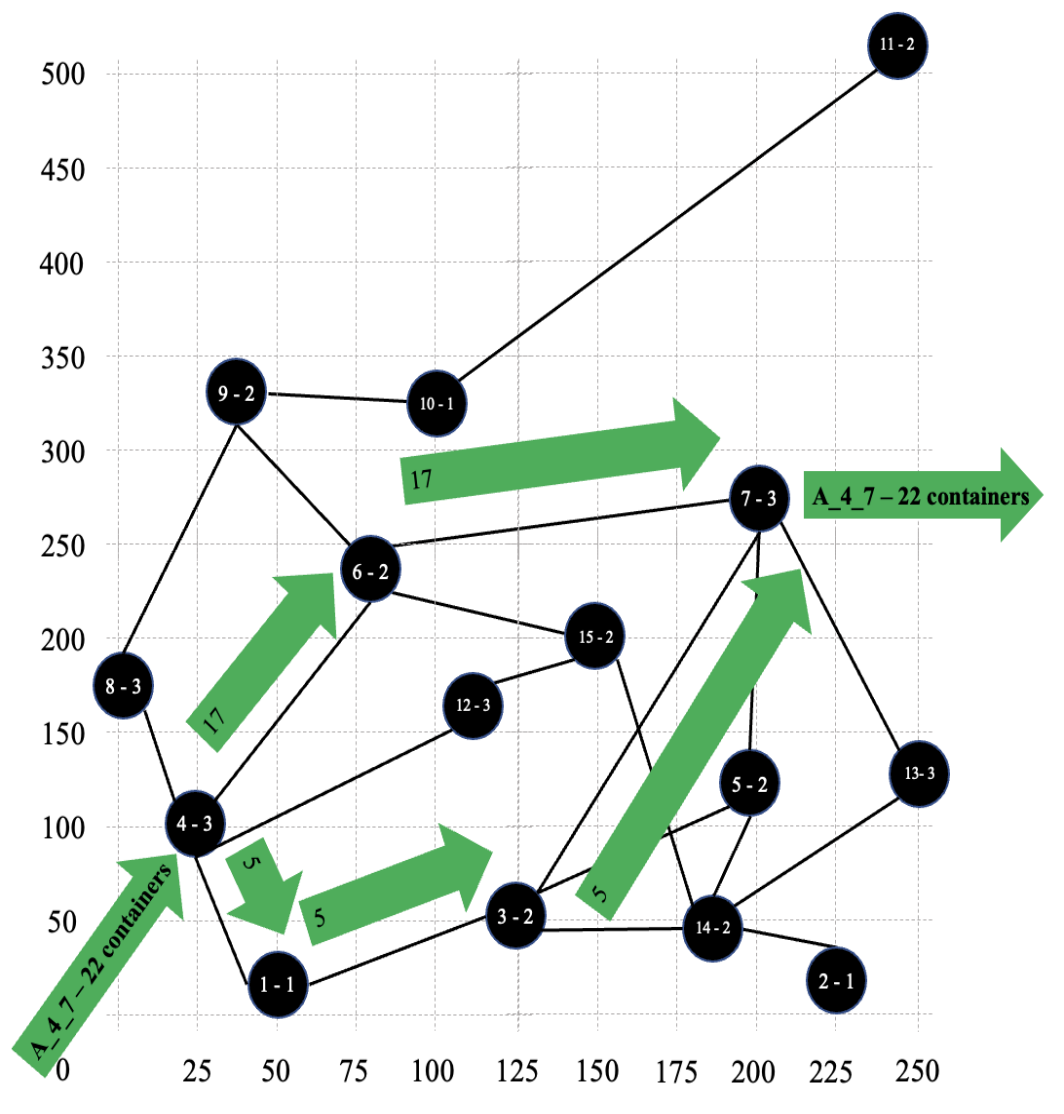


Figure 3. Network Flow for Company A_4_7

For the shipment from Company B going from station 1 to station 11, all 8 of the containers along the same path. This path goes from stations 1-4-8-9-10-11 (see Figure 4, Table 5). Despite going through numerous stations, this is actually the best possible routing. The only alternate route that is a candidate would be 1-4-6-9-10-11, but that has a greater distance and the same number of stations, making it worse. Therefore, the 8 containers going from 1-4-8-9-10-11 is the most cost effective possible route for this shipment.

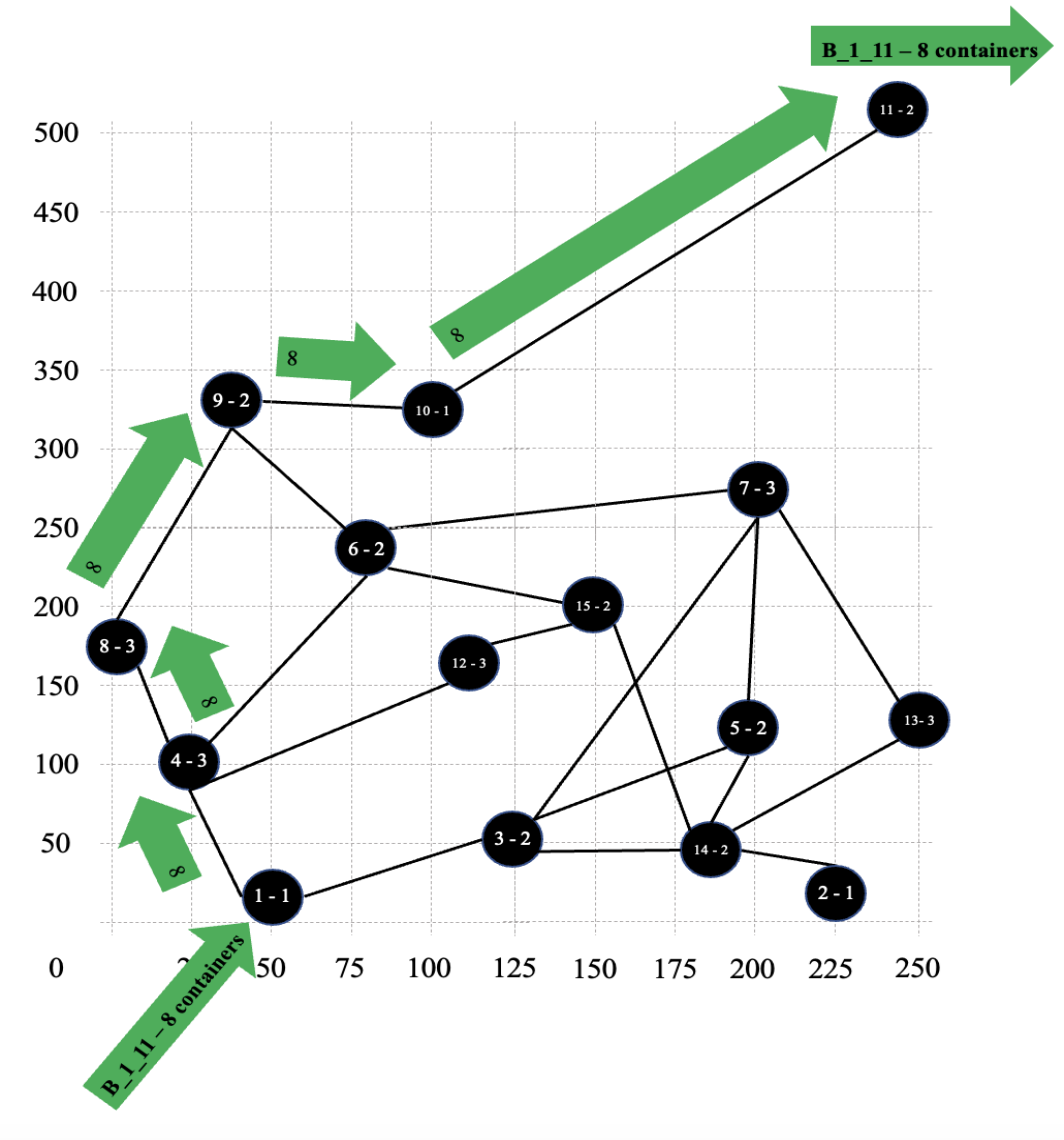


Figure 4. Network Flow for Company B_1_11

For the shipment from Company C from station 13 to station 1, the 7 containers are split into two different paths. 3 of the containers take the most direct route from station 13-14-3-1. The remaining 4 containers go along the path 13-7-3-1 (see Table 5). Of interest here is that the containers diverge in their routing from station 13 to station 3, but all 7 of the containers go on the same routing from 3-1. As can be seen in Figure 5 below, the most efficient routing between stations 13 and 3 is 13-14-3 (station 7 is out of the way). However, the reload capacity constraint at station 14 causes four of the containers to be rerouted in a more costly manner.

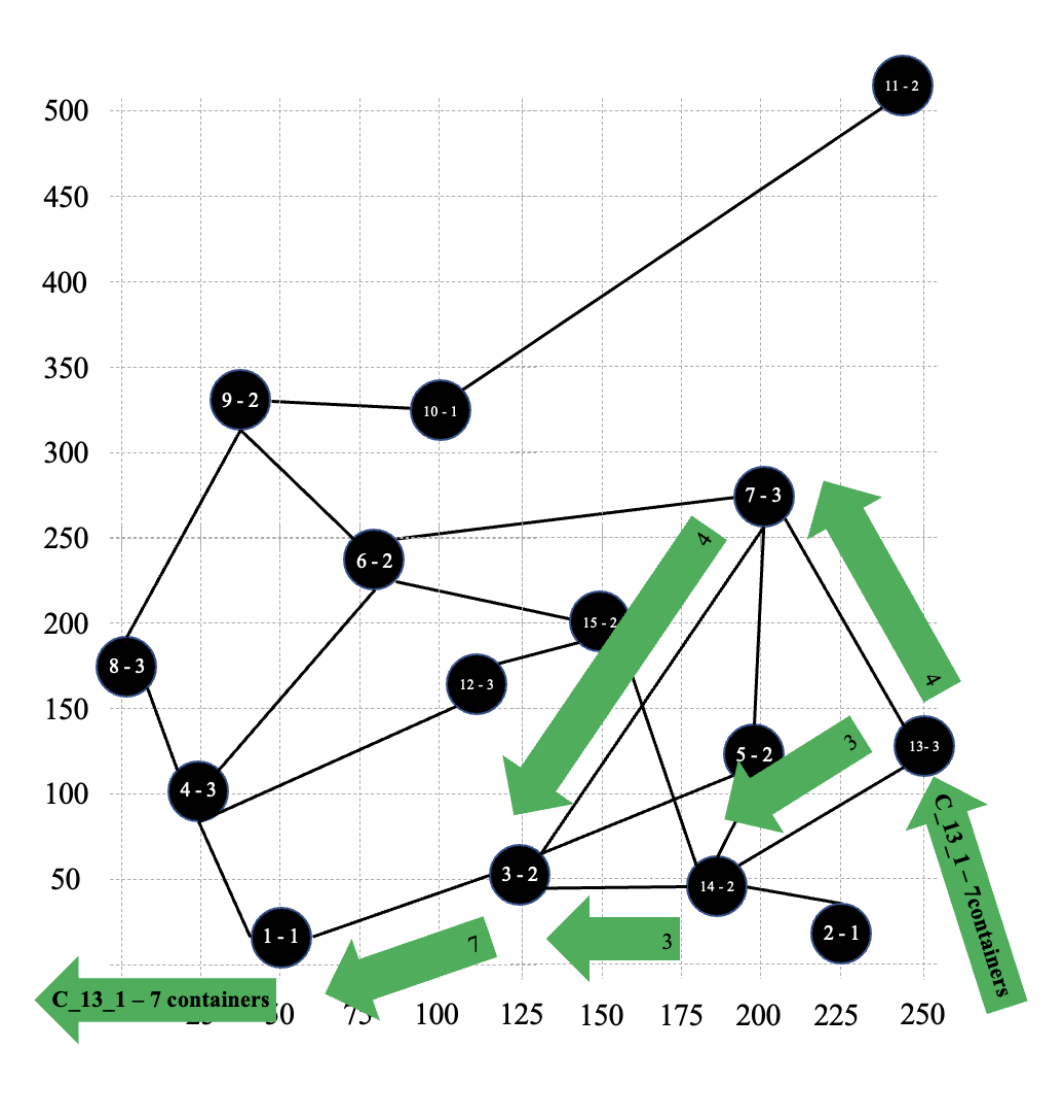


Figure 5. Network Flow for Company C_13_1

For Company C's shipment from station 13 to station 7, both of the 2 containers go on the same path. That path is simply from station 13-7 (see Table 5). This direct path is trivially the most efficient route for moving these two containers (see Figure 6), minimizing the distance and number of reloads necessary.

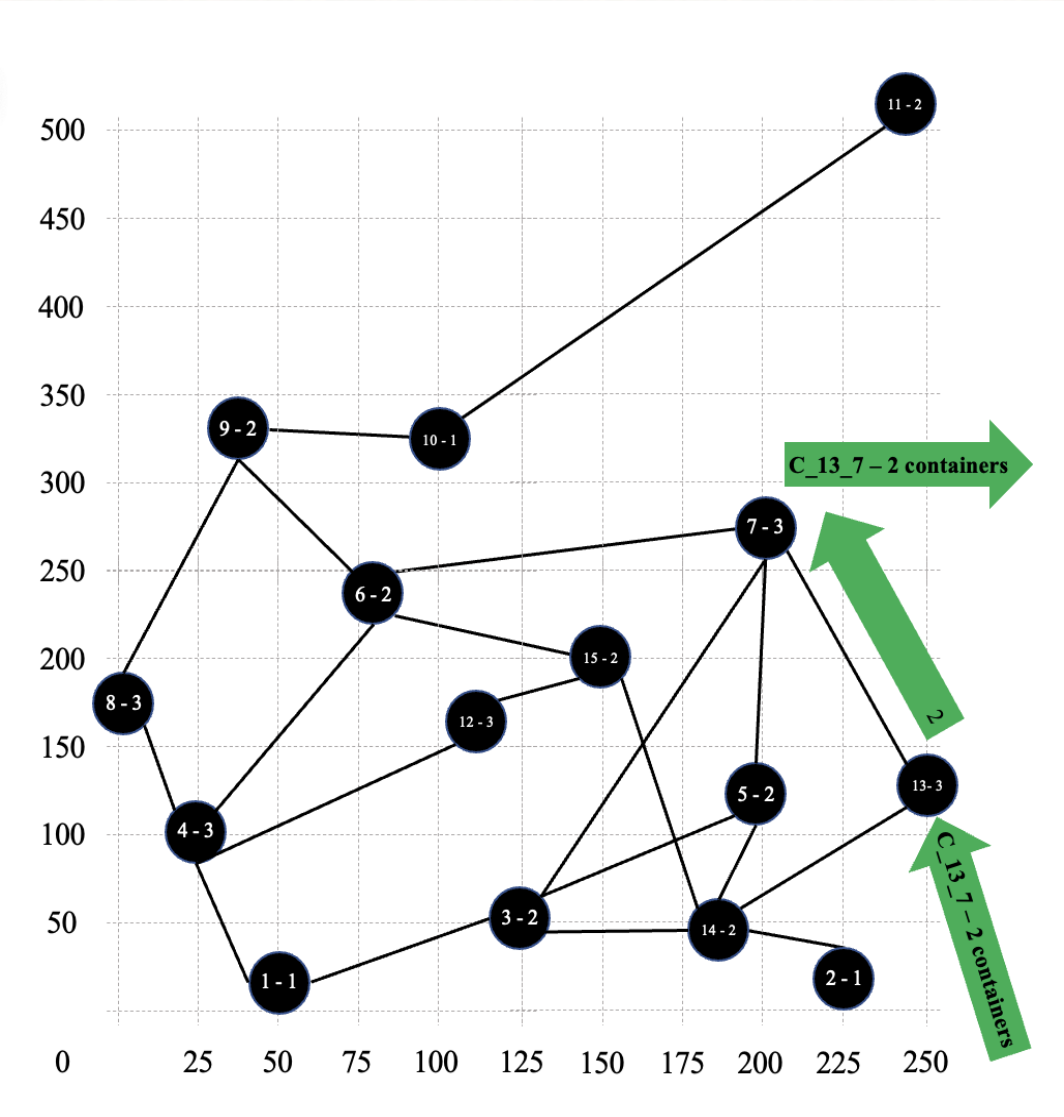


Figure 6. Network Flow for Company C_13_7

For the shipment belonging to Company C from station 13 to station 8, the single container goes along the path 13-14-15-12-4-3 (see Table 5). This may seem long, but as can be seen in Figure 7 below, the only other path that seems reasonable is 13-14-3-1-4-8. However, reload capacity constraints at station 1 prevent this from being a possible routing. Therefore, 13-14-15-12-4-3 is the next best option for moving this single container.

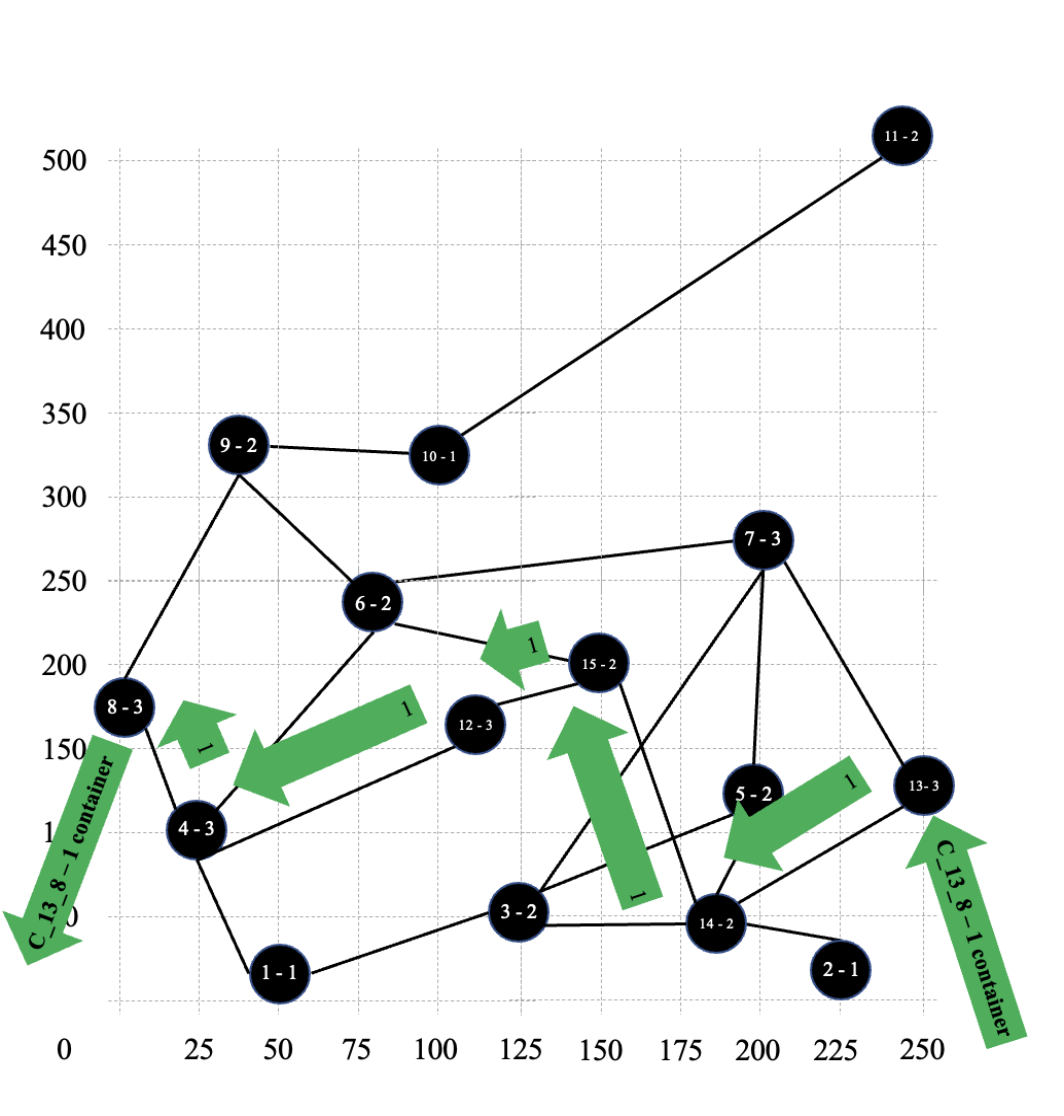


Figure 7. Network Flow for Company C_13_8

For Company D's shipment from station 5 to station 6, the 6 containers were shipped on two different paths. 3 of the containers took the route 5-14-15-6 and 3 of the containers took the route 5-7-6 (see Table 5). As can be seen in Figure 8 below, the most efficient routing for the containers was 5-7-6. The reload capacity limit at station 7 as well as the track limit constraint along track 6-7 contributed to the need to split the containers into two routes – one more efficient than the other.

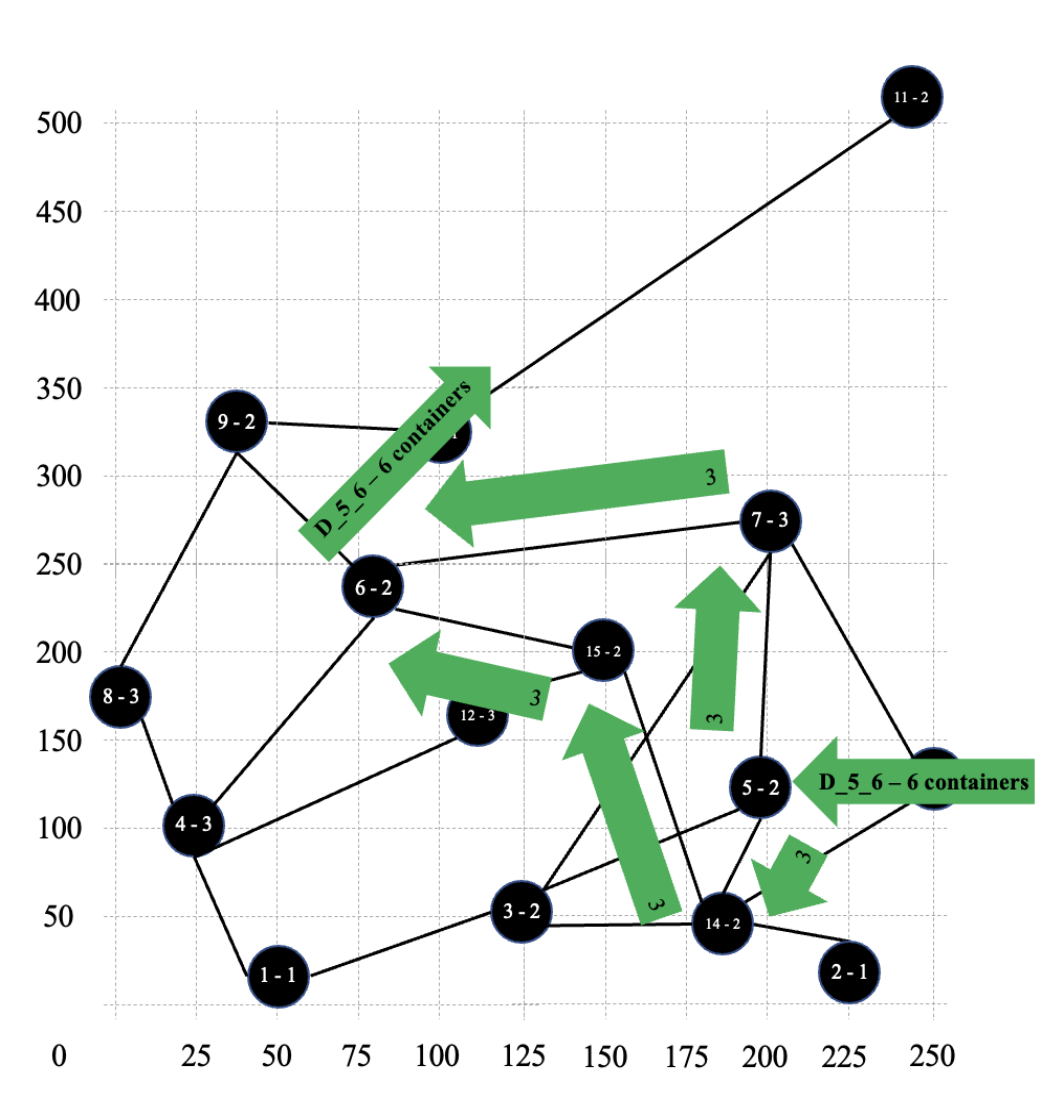


Figure 8. Network Flow for Company D_5_6

For the shipment from Company D from station 5 to station 10, the 2 containers took different routes. The first container took the route 5-14-15-6-9-10 while the second took the route 5-14-15-12-4-8-9-10 (see Table 5). As can be seen visually in Figure 9, the shortest route with the fewest number of stations would be 5-7-6-9-10. However, reload limits at stations 6 and 7 as well as capacity constraints along track 6-7 led to the need for the two inefficient, split routing.

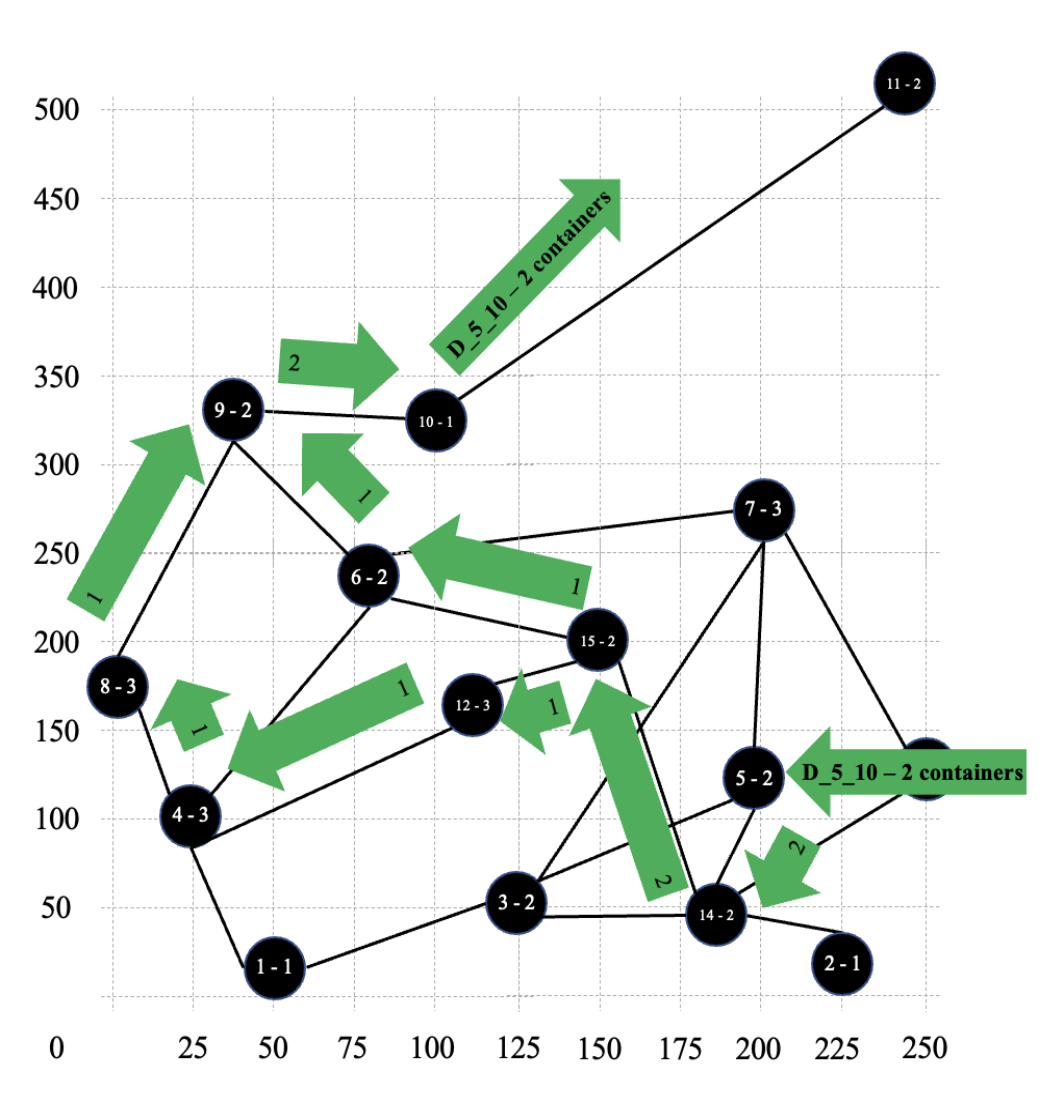


Figure 9. Network Flow for Company D_5_10

For Company E's shipment from station 3 to station 13, the 4 containers all go along the same path. That path is stations 3-7-13 (see Table 5). However, as can be seen in Figure 10, the most efficient route would be to go from station 3-14-13 (that path is much shorter). However, reload limits at station 14 cause the need for this split routing.

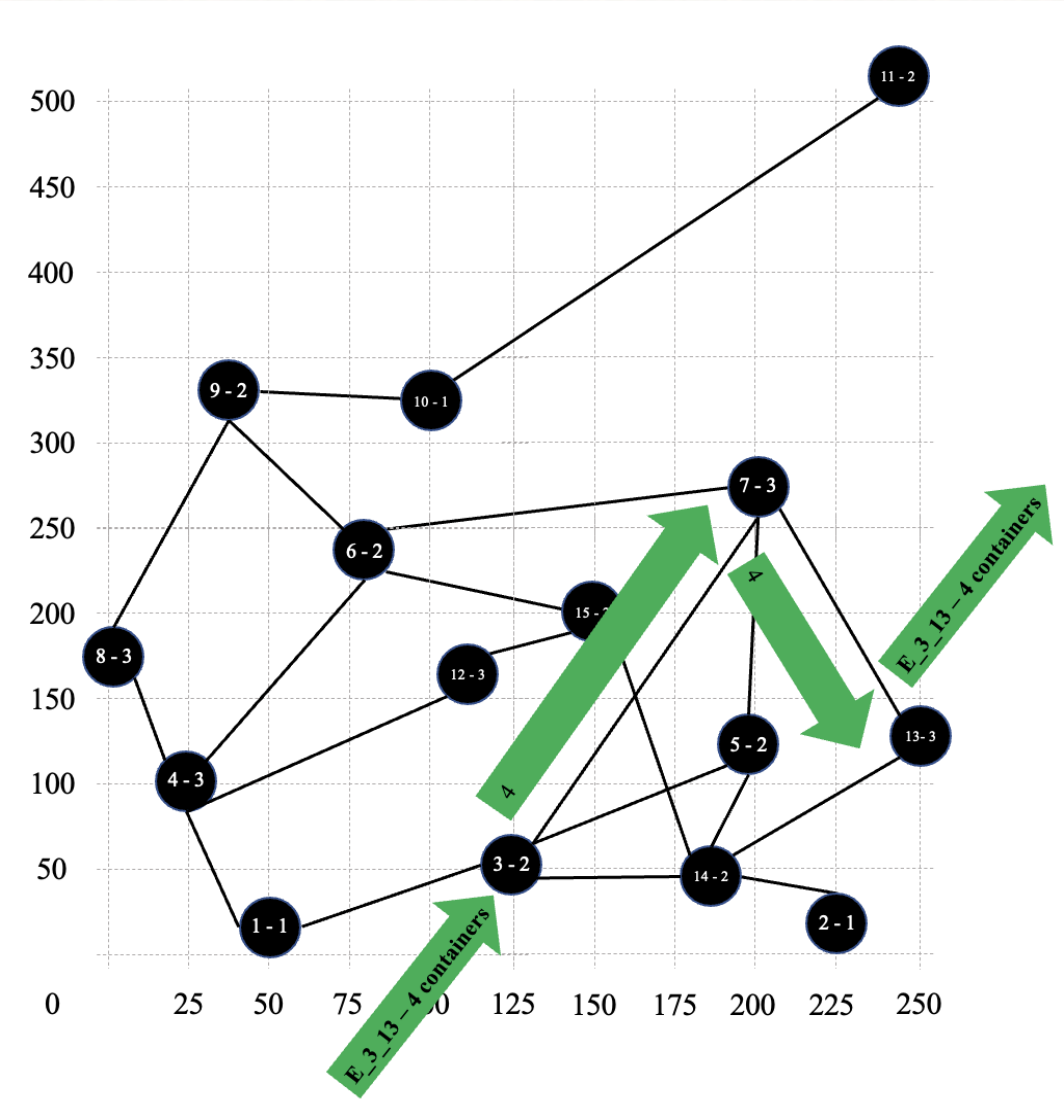


Figure 10. Network Flow for Company E_3_13

Finally, for Company E's shipment from station 3 to station 15, all 10 of the containers go along the same path. That path is stations 3-14-15 (see Table 5). As can be seen in Figure 11 below, this routing is also the shortest route between stations 3 and 15 (there is no track 3-15 or 5-15). Therefore, these 10 containers are moved in the most efficient manner possible based on SmartRail's map.

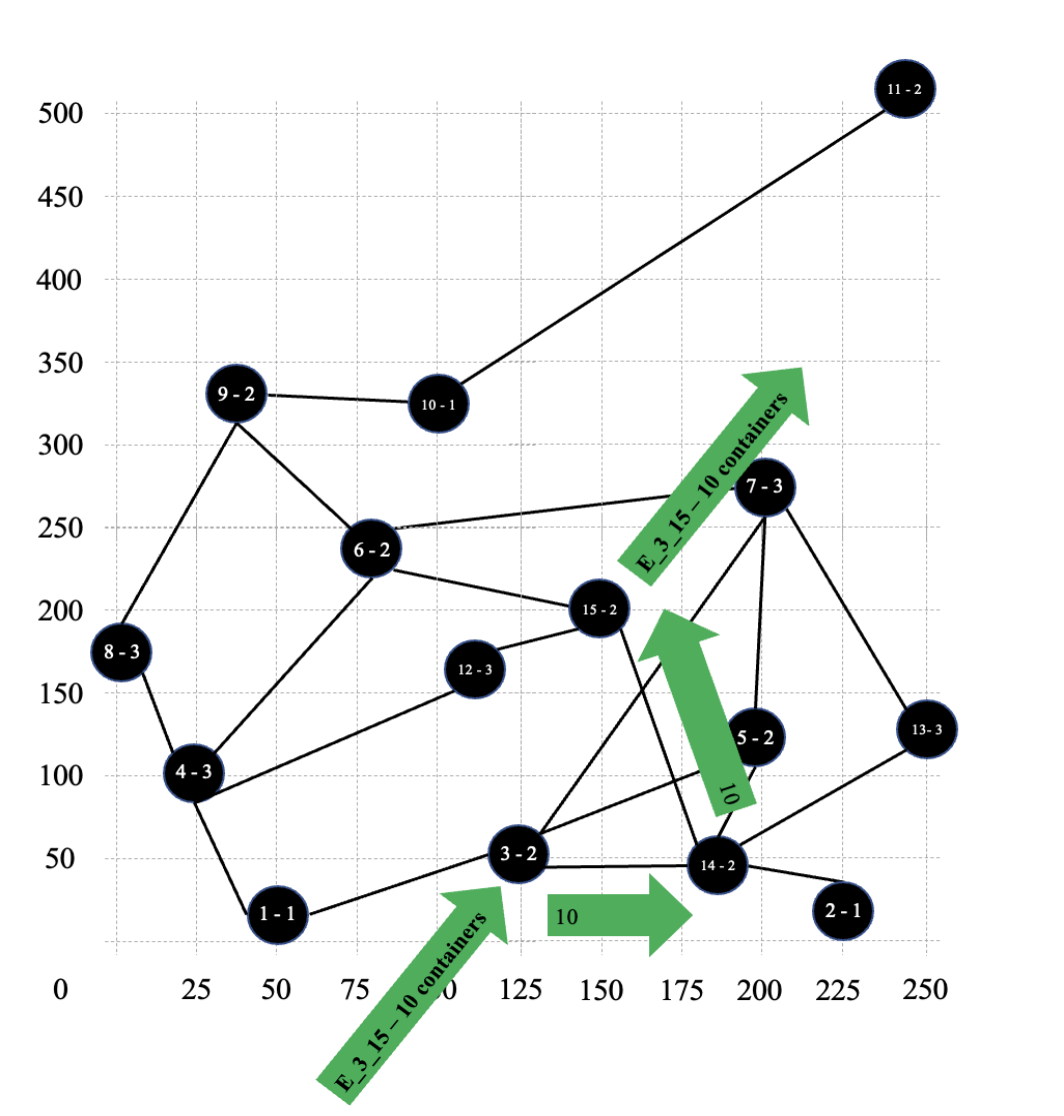


Figure 11. Network Flow for Company E_3_15

In the analysis for each of the shipments, we discussed the best route (assuming no capacity limits on reloads or track), but in some cases they were constrained. We note that in a world without track limit or reload zone limits, all containers moving along the same route would be the most optimal (unless two routes have the exact same distance and have the same number of reloads, in which case it is an alternatively optimal solution).

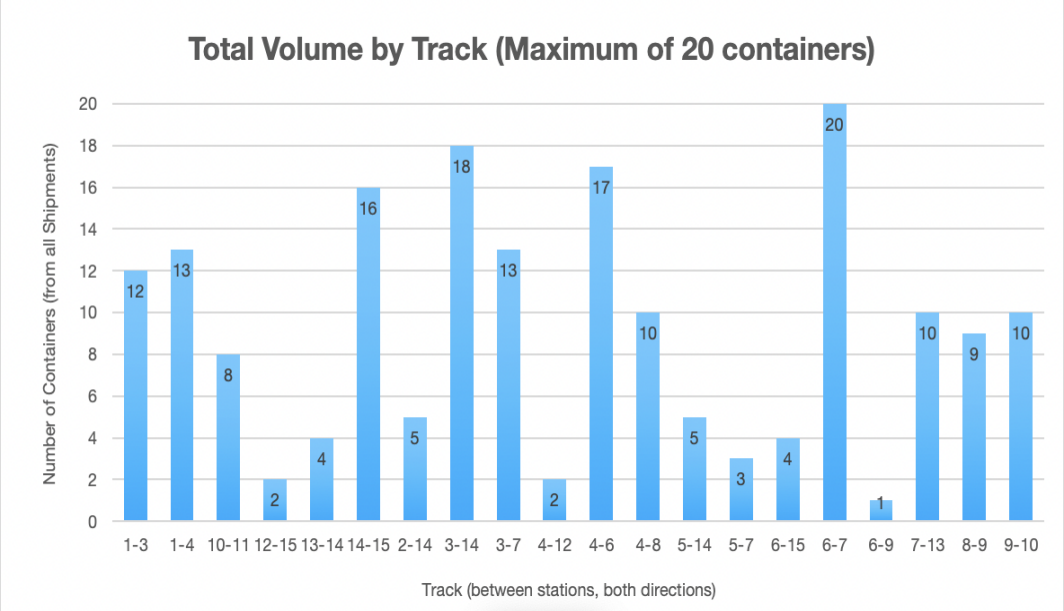


Figure 12. Total Volume by Track (Maximum of 20 containers)

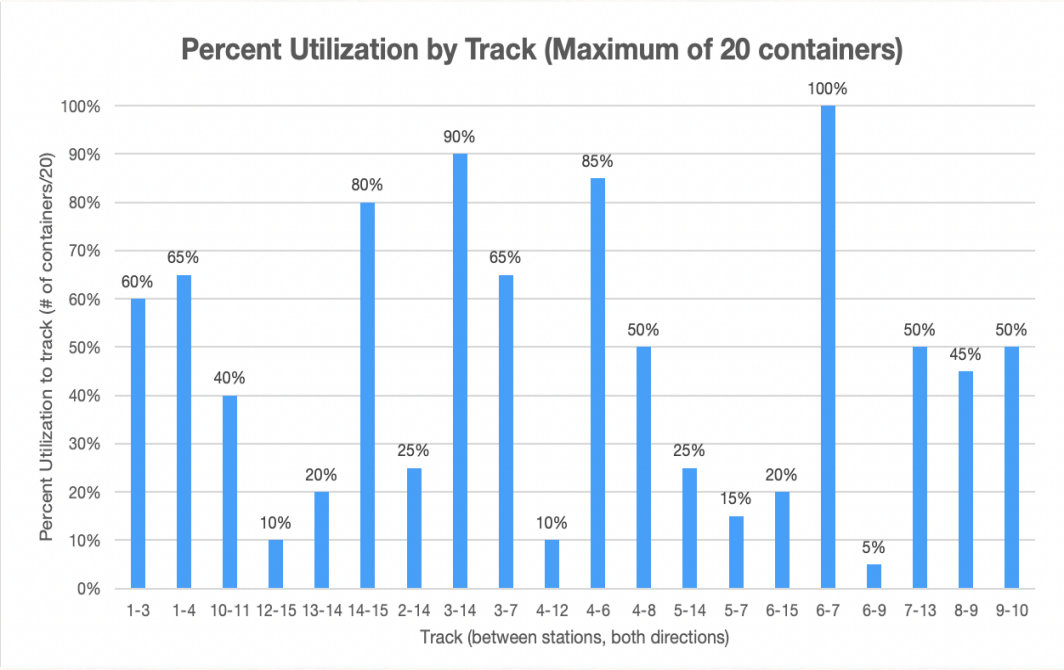


Figure 13. Percent Utilization by Track (Maximum of 20 containers)

Figure 12 & 13 were calculated from the optimal solution of the model to look at the track utilization with respect to track limits (Note, figures generated in an excel file from the SmartRail-CompletePlan.csv output from the Task 2 AMPL code). Figure 12 shows the number of containers that move along a given track, while Figure 13 shows the percent utilization of each of the tracks. Based on the two graphs, the only track that is used at its maximum capacity is the track from station 6 station 7. As described in the analysis above for each company, this was one contributing factor in the need to split the paths for shipment A_4_7 as well as both shipments from Company D. In the future, SmartRail could consider expanding the capacity along this track to attempt to reduce costs (assuming the savings are larger than the expansion costs).

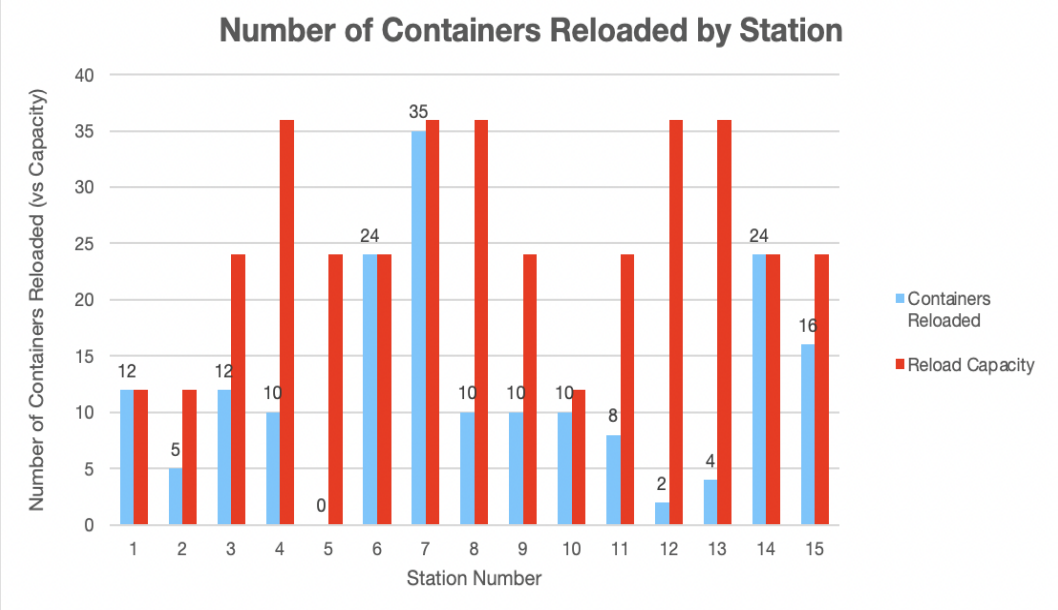


Figure 14. Number of Containers Reloaded by Station

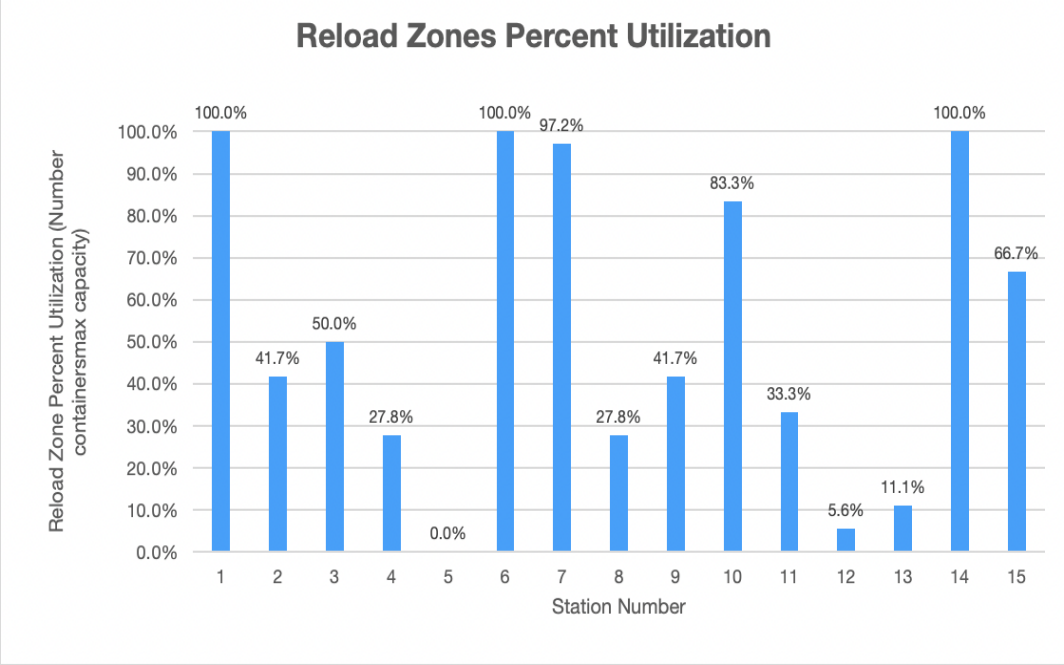


Figure 15. Reload Zones Percent Utilization

Figures 14 & 15 were calculated from the optimal solution of the model to look at the number of containers loaded at each station as well as the percent utilization for the reloading zones (Note, figures generated in an excel file from the SmartRailCompletePlan.csv output from the Task 2 . Figure 14 shows the total number of containers that are reloaded at a station (with the maximum capacity listed next to it). This is summarized in Figure 15 which shows the percent utilization of each of these reload zones. As can be seen in Figure 14 & 15 above, a majority of the stations have a very small amount of their reload capacity utilized. That being said, three stations (station 1, 6, 14) have 100% of their capacity used and station 7 has 97.2% of its capacity used. The implication of this is in the fact that some containers had to go along a less efficient route relative to the constraints. Specifically, shipments A_4_7 and D_5_10 have less efficient routing due to the station 6 reloading constraints. The two shipments from company D are both less efficient than theoretically possible due to the station 7 reload limits. The reload capacity constraint for station 14 causes the shipments E_3_13 and C_13_1 to go on a path that is not the most cost effective. Finally, the capacity being maximized at station 1 prevents shipment C_13_8 from taking the best possible route. Therefore, in the future, SmartRail should consider expanding the number of reloading zones at stations 1, 6, 7, and 14. It should be noted that expanding reloading zones may also require track capacity to expand to reach the most efficient routing (though the cost of doing so may not offset the savings - so it may not be most optimal).

There is one additional point of interest to be made with respect to the optimal solution. In the programming of this final solution, the team discovered that the final solution is not actually unique. The minimum objective value (minimum costs) are $3,706,028.92, however there are multiple ways to do that. One specific example has to do with the movement of containers from station 13 to station 3. The shortest route is stations 13-14-7. However, the reload capacity is at its maximum for station 14, causing some containers to be rerouted 13-7-3. Within the shipments C_13_1 and E_3_13, 8 containers move along the route 13-7-3 (they go in different directions, but that does not matter here) while 3 go 13-14-3 (see Table 5, Figures 5 and 10). However, there are multiple ways to split this up amongst C_13_1 and E_3_13. This leads to multiple possible paths that are all optimal. So, SmartRail could make adjustments to their schedule to make their life easier (such as it might be easier to keep shipment C_13_1 all together and split up E_3_13; the opposite is achieved in our project). Despite all of this, SmartRail should realize that if their goal is simply to minimize costs, these differences in container paths do not matter. They do not matter because the SAME minimum cost is achieved, as long as the solution is an OPTIMAL solution. Therefore, SmartRail should have no concern implementing the solution our team presented, as there is NO other feasible solution that can lower the costs more.

In summary, the solution results in a minimized cost of $3,706,028.92 for moving the ten shipments from their origin to the destination.. While some are moved in the most efficient manner possible, the reload and track capacity constraints lead to some containers being routed in a more costly way, such that the MILP outputs an optimal solution that is FEASIBLE. Thus, SmartRail could consider doing a cost-benefit analysis in the future to see if expanding the track capacity (by adding more tracks) or number of reload zones could further minimize costs in the future. In the future, SmartRail could also consider building new tracks to connect previously unconnected stations (such as 3-13), again requiring a cost-benefit analysis to see if the cost savings offset the cost of expansion. All that being said, the routing described in Table 5 above and shown through each of the figures, is as good or better than any other feasible solution, and therefore SmartRail can implement this plan to ship the containers at the lowest possible cost.

4 AMPL Instructions for Client

The following section contains a detailed set of instructions for client usage of the AMPL set up. It also includes a brief explanation for interpreting the AMPL and .csv output.

4.1 Running AMPL with CSV Files

The first step for SmartRail is to download AMPL onto a computer, which can be done using the instructions here. The AMPL files should be placed into a folder on the computer that is easily accessible (such as Desktop or Documents). Once AMPL is successfully installed on the computer, the client can then download the SmartRail_ScheduleSoftware folder, which the team has submitted to the client. This SmartRail_ScheduleSoftware folder should be in an easy to find location on the client's computer (such as Desktop or Documents). Inside the AMPL IDE (Amplide), there are three windows. The leftmost window corresponds to the directory, which should be adjusted (by clicking through the file path) to be SmartRail_ScheduleSoftware. The middle window is the console window, which we will use later. The rightmost window will be used at the end for viewing the .log results file.

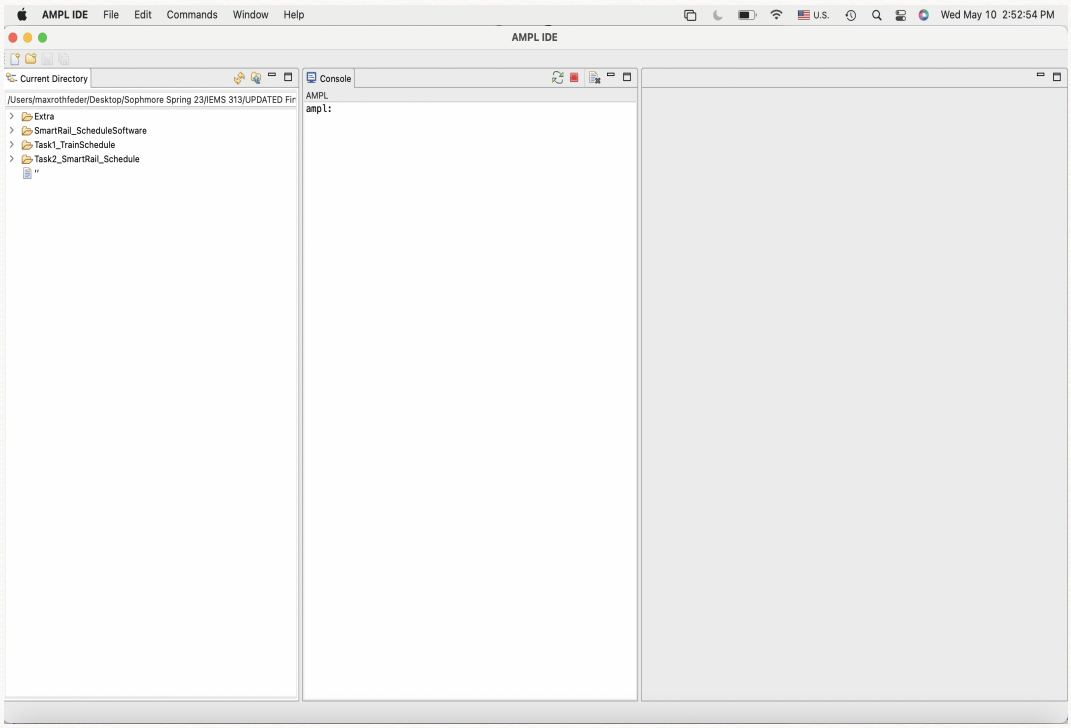


Figure 16. AMPL Initial View

The SmartRail_ScheduleSoftware folder has two files in it to begin with: SmartRailTrainSchedule.mod and SmartRailTrainSchedule.run. These should be visible on the leftmost window

of AMPL, if the SmartRail_ScheduleSoftware folder has been clicked on as the current directory.

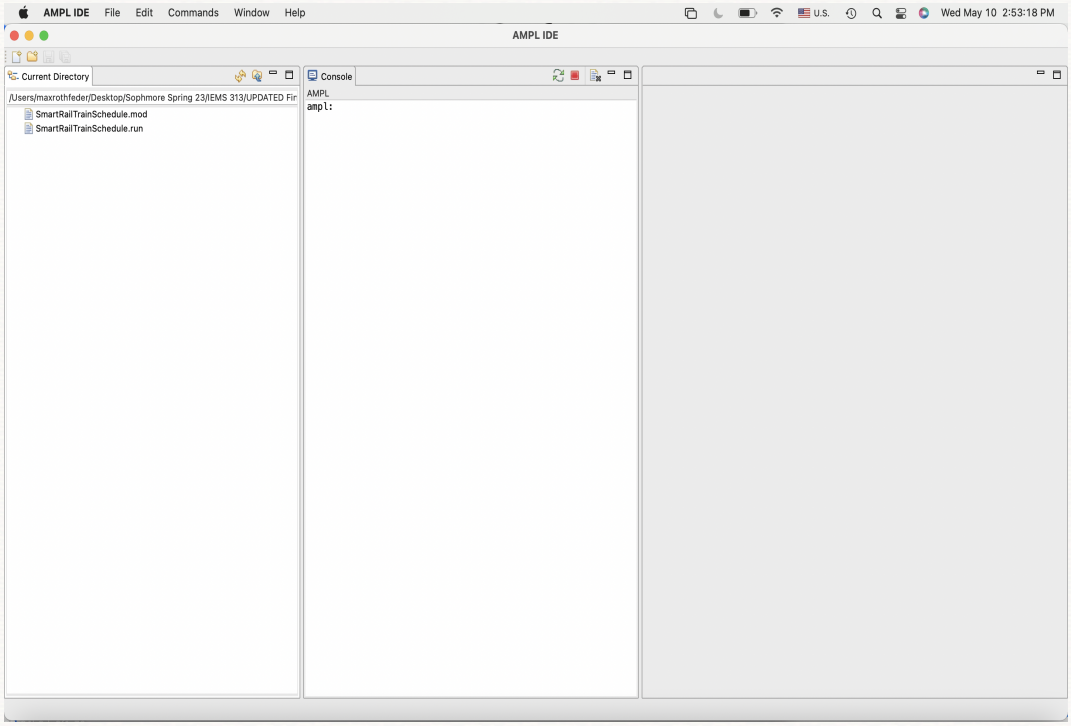


Figure 17. AMPL Selection of Directory

Note: The version submitted to the client for this project contains the .csv files and a results (.log) file. The instructions below will explain to the client how each of these files were created and can be updated in the future (and also how to redownload the plug-in for their own computers).

The next step is ensuring that SmartRail's computer has the AMPL csv file downloaded. This can be done by going to the website attached here and downloading the package corresponding to the client's computer type (MacOS, Linux, Windows). Once downloaded, the amplcsv.dll file should be dragged into the ampl folder (ampl.exe on windows). The image below shows this ampl folder, with the amplcsv.dll file visible in the left window of AMPL.

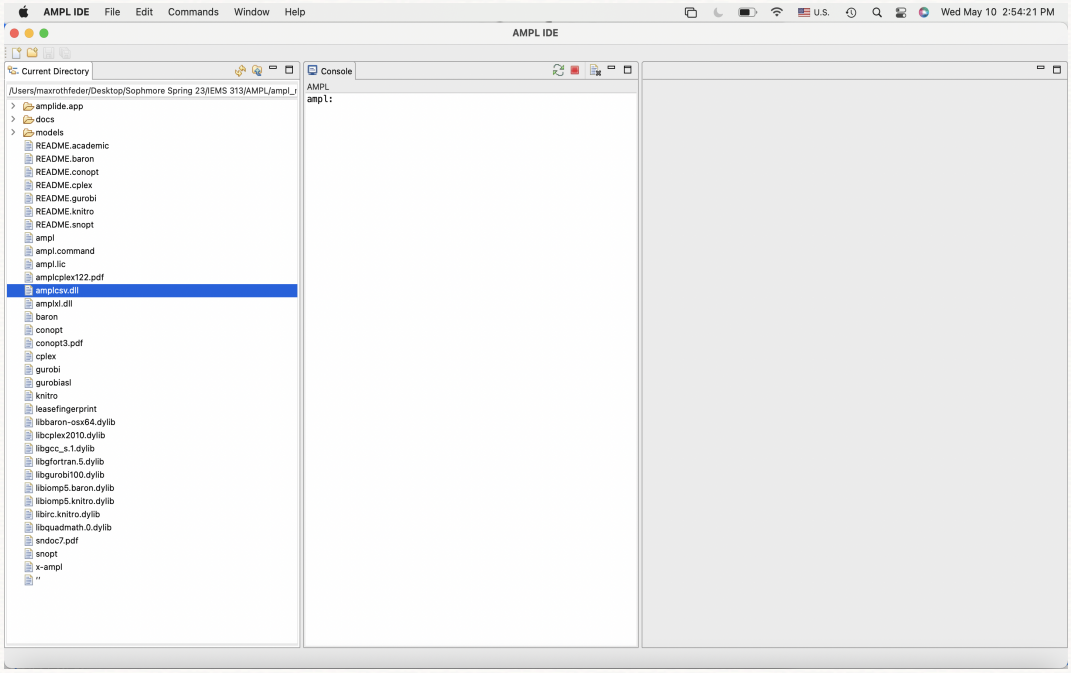


Figure 18. AMPL Addition of Folder and Selection of File

The last step is for SmartRail to add in their .csv files. First, navigate back into the SmartRail_ScheduleSoftware. As per the description given to our team, there will be four files other_param.csv, shipments.csv, stations.csv, and tracks.csv. While the client can change the data within these files over time (including adding new shipments, stations, or tracks), the format and column titles should remain constant. SmartRail should download these four files and place them into the SmartRail_ScheduleSoftware folder, ensuring that their names do not change. They should now be visible on the left window of AMPL (the window may need to be refreshed by clicking the two yellow arrows in the directory window).

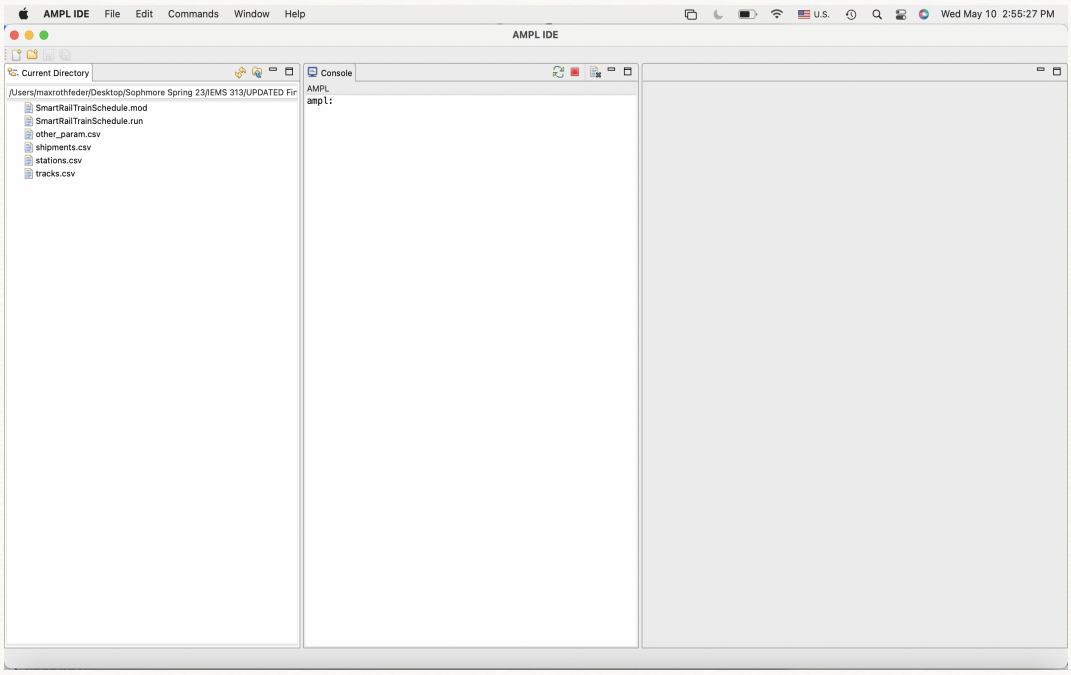


Figure 19. AMPL Addition of .csv Files

Note: If SmartRail wants to make edits to the schedule, they can either edit within the .csv files or remove the current .csv file (by deleting) and adding in an updated version of the .csv file (with the same name).

Now, the software is ready to be run. In the central "Console" window, the following code should be typed, followed by the "Return" or "Enter" key on the keyboard:

```
include SmartRailTrainSchedule.run;
```

It is imperative to NOT forget the semicolon at the end of this statement.

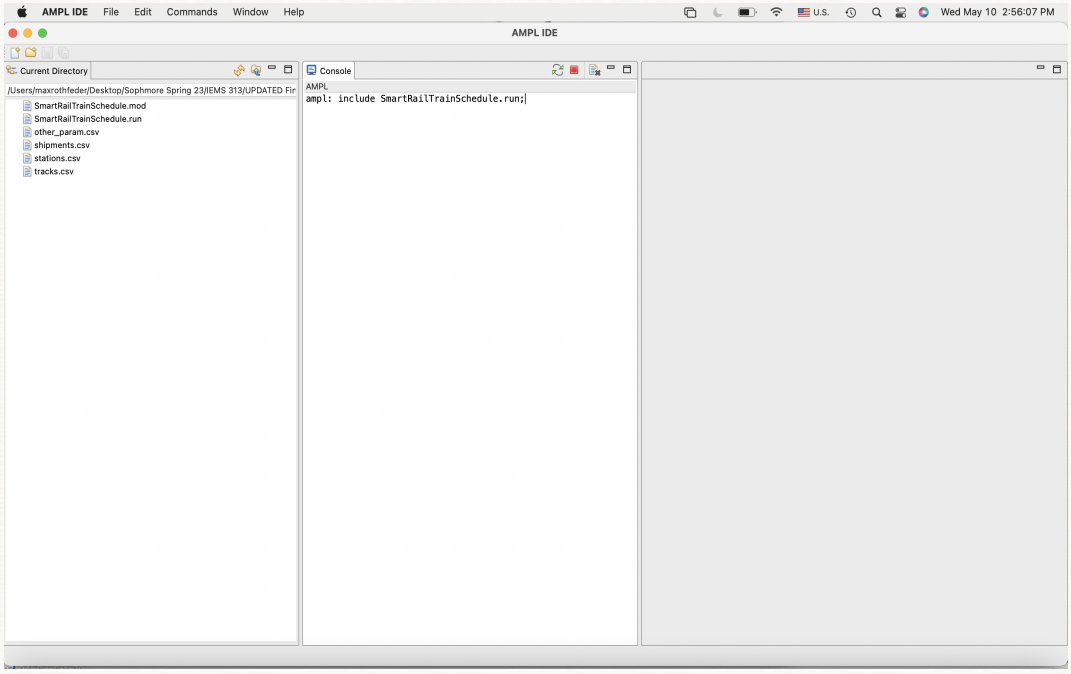


Figure 20. AMPL Program Initialization

Once this runs, a solution should be printed in the center window, and once refreshed, a file SmartRailTrainSchedule.log will appear in the left window. This contains the solution.

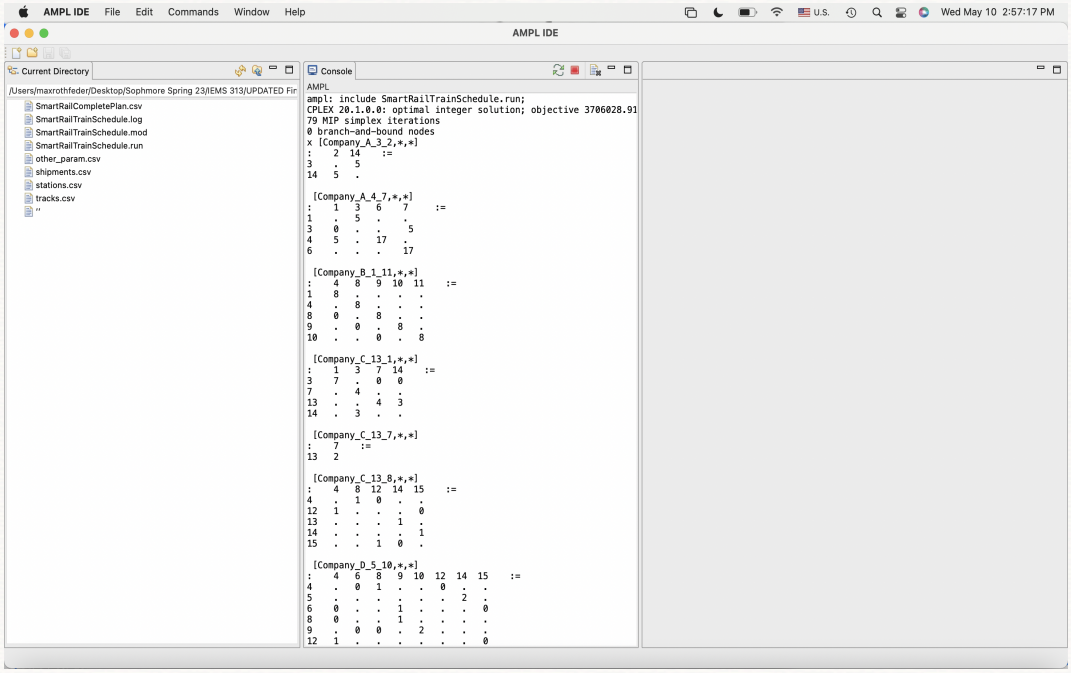


Figure 21. AMPL Program Output

4.2 Interpreting the Results

The easiest way to view the results is by clicking on the SmartRailTrainSchedule.log file, which should open up in the rightmost window of AMPL.

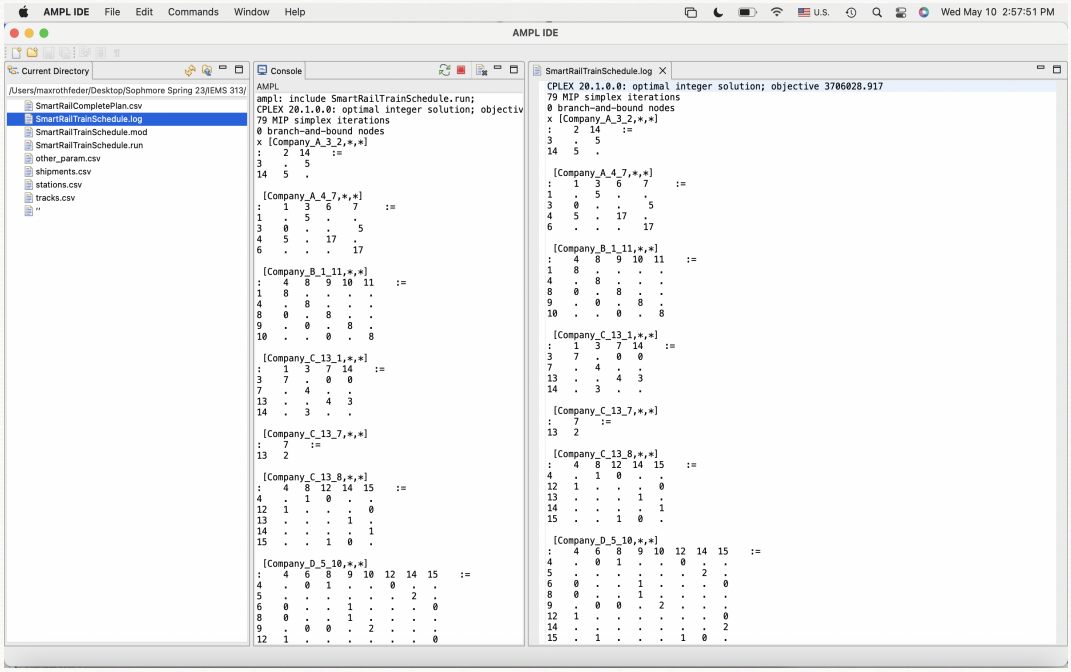


Figure 22. AMPL SmartRailTrainSchedule.log File View

The first line in this file says objective: 3706028.917. That means the minimum cost achieved for running this specific train schedule is $3,706,028.92. With different .csv files (corresponding to a different network), this result will likely change.

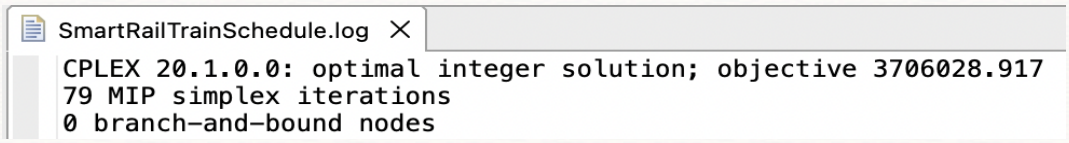


Figure 23. AMPL SmartRailTrainScheduling.log File View Magnification

The next key is to interpret the results tables. For simplicity, the team has coded the file to not include any rows/columns within the matrix that does not contain any relevant information for the user. A separate table will be created for each individual shipment, describing how the shipment should be moved. One such example is included in Figure 24 below. The key to reading this table is to realize that the numbers on the left correspond to the starting station and the numbers on the top correspond to the ending station. Then, the client simply should begin with the starting station (in this case 4) and follow each of the paths until the ending station (in this case 7). Each container must make it from the start to the end, so the number of ways containers are moved dictate the number of paths. In this case, two paths are followed. The numbers within the matrix dictate how many containers are moved along that track.

Path 1: Station 4 -> Station 1 -> station 3 -> Station 7 (with volume of 5)
Path 2: Station 4 -> Station 6 -> Station 7 (with a volume of 17)

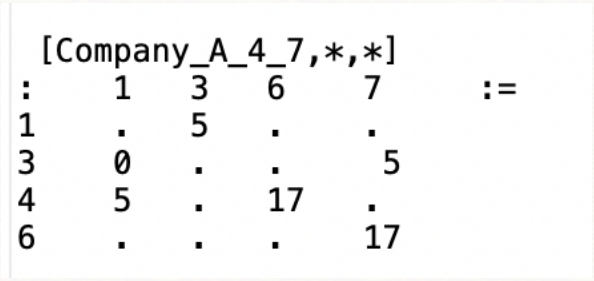


Figure 24. AMPL Table Example

In addition to the log file, our team felt that SmartRail would also enjoy receiving all of the information in decision variables. Looking back at the AMPL interface, there is also a file called SmartRailCompletePlan.csv.

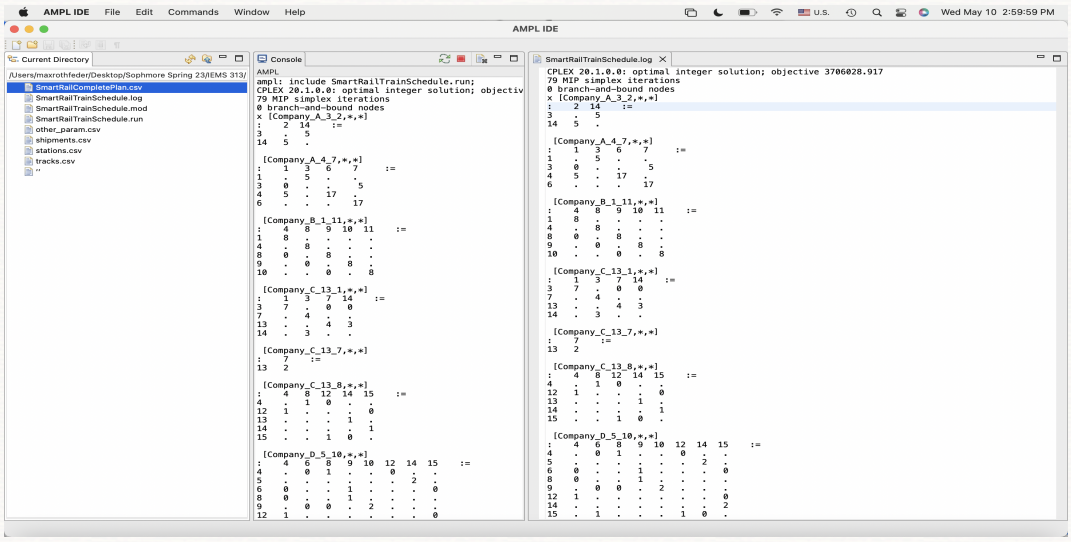
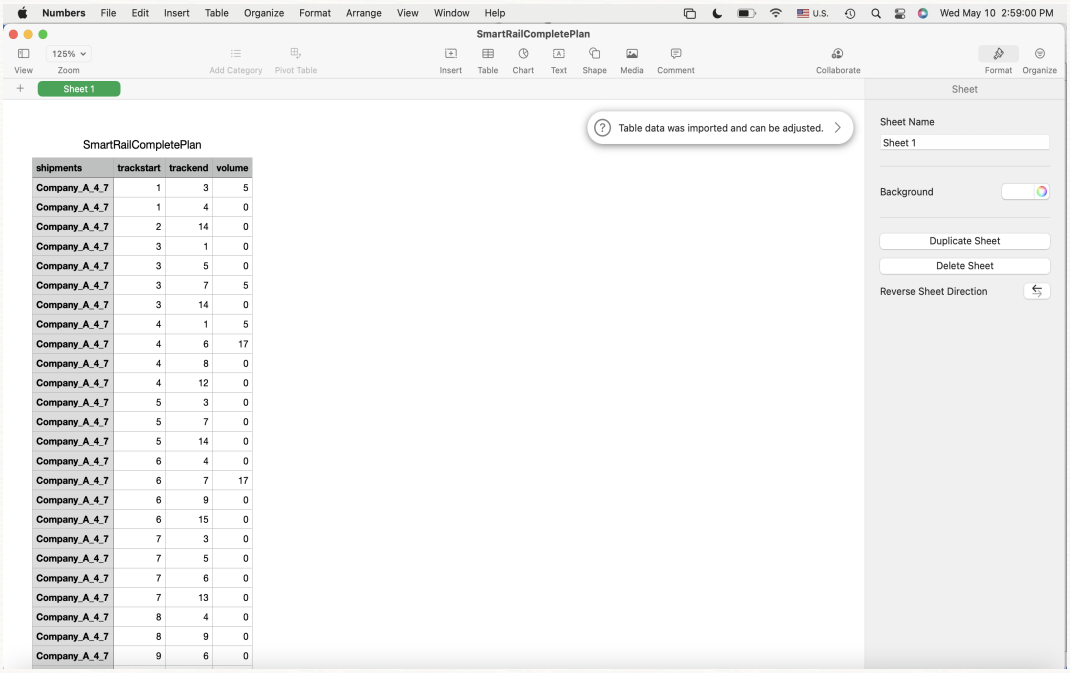


Figure 25. AMPL SmartRailCompletePlan.csv File Selection

By double clicking on this file, a window will pop up (in an application depending on the computer type) displaying the .csv file of all the decision variables. It lets SmartRail know the shipment, track (trackstart, trackend), and the volume moving along the routing. It includes all decision variables (including those that are zero), but a quick sort in Excel will remove those. This information could be helpful for SmartRail as they analyze the data (for example, the charts the team made in Analysis 3.2 were created from this.csv file).



shipments	trackstart	trackend	volume
Company_A_4_7	1	3	5
Company_A_4_7	1	4	0
Company_A_4_7	2	14	0
Company_A_4_7	3	1	0
Company_A_4_7	3	5	0
Company_A_4_7	3	7	5
Company_A_4_7	3	14	0
Company_A_4_7	4	1	5
Company_A_4_7	4	6	17
Company_A_4_7	4	8	0
Company_A_4_7	4	12	0
Company_A_4_7	5	3	0
Company_A_4_7	5	7	0
Company_A_4_7	5	14	0
Company_A_4_7	6	4	0
Company_A_4_7	6	7	17
Company_A_4_7	6	9	0
Company_A_4_7	6	15	0
Company_A_4_7	7	3	0
Company_A_4_7	7	5	0
Company_A_4_7	7	6	0
Company_A_4_7	7	13	0
Company_A_4_7	8	4	0
Company_A_4_7	8	9	0
Company_A_4_7	9	6	0

Figure 26. Excel Improved Visualization

The team has submitted two files to SmartRail. One file - Task2_SmartRail_Schedule includes all of these log, run, csv (input and output), and mod files in it for the problem requested for our team to solve. This contains the solution to the question that SmartRail wanted solved. The other file - SmartRail_ScheduleSoftware enables SmartRail to start from scratch and input the .csv files manually by following the instructions above. They could also just delete the .csv files from Task2_SmartRail_Schedule, once they've downloaded the .csv plugin.

5 References

Section	Description	Author
5.1	AMPL Course Package Installation Instructions	AMPL Optimization Inc.
	Link	
	https://ampl.com/licenses-and-pricing/ampl-for-teaching/ampl-course-install/	
Section	Description	Author
5.1	AMPL CSV File Interface	AMPL Optimization Inc.
	Link	
	https://amplplugins.readthedocs.io/en/latest/rst/amplcsv.html	