1994: The Communications Network Problem

Ashlyn Orr & Isaac Dugan

Math 465: Mathematical Models and Applications

Dr. Steven Hetzler

December 18, 2023

Executive Summary

Summary for situation A:

For situation A, we are attempting to find the fastest possible way to transfer files between departments. Each department can only participate in one transfer at a time and each transfer takes 1 minute. We were able to devise an algorithm that goes through the departments and transfers and creates an optimal schedule. Because some of the departments are involved in three transfers and only one transfer can occur at a time, the fastest all transfers can be completed is 3 minutes. You can see the schedule we created in Figure 1 (each color corresponds to a different minute). We have written code to perform this algorithm, so it will be very easy to find a new optimal schedule if we change the number of departments or what transfers must occur.

Summary for situation B:

In this situation, we have computers and transfers that must occur (building off of situation A), but the transfers now take different amounts of time. By modifying our algorithm from situation A, and using a technique called a dependency graph, we were able to find a schedule that is optimal and takes 23 minutes to complete all transfers between departments (you can see the order in which the transfers should take place in figure 5). We will have to do some further work to confirm that our process will find an optimal schedule for any combination of transfers and departments. Transfers 14 to 18, 14 to 20, and 20 to 24 are bottlenecks, so we should look for ways to decrease the time those specific transfers take.

Summary for situation C:

In this situation, we built on situation B by adding some additional transfers and upgrading some of the department's computers to be able to handle multiple transfers at once (the actual number varies by department). We were able to find an optimal solution that will complete all transfers in 29.6 minutes (see figure 8). Department 24 must complete 5 transfers and its computer can only handle one at a time. A computer upgrade for this department would speed up the entire transfer process. Unfortunately, we were unable to create a scheduling framework that would work for any combination of departments, transfers, and computer capacities. If we were to change any of these parameters, we would have to go back to the drawing board and find a new optimal transfer schedule.

Restatement Of Problems

This problem is asking you to explore a company's network as they transfer data between different departments. There are three situations in which they are asking you to consider. For Situation A, we were asked to find the shortest possible schedule for a given network. The network involves 28 departments, represented by the vertices (with 27 edges connecting each). These edges are representing the transfers that need to occur each day. For Situation A we are given that each transfer takes the same amount of time, 1, and that each edge represents 1 transfer. Our job is to find the shortest possible schedule to have these files transferred. A visual of this network can be found in Figure 1 of the appendix.

For Situation B we are following most of the same criteria from Situation A. The difference between these scenarios is that in Situation B, each department has a specific time that it takes to transfer their files to a neighboring department. These times are shown in Table 1 of the appendix. We must create an algorithm that allows these networks to send files in the shortest time possible. We also want to conclude if this solution would work for most cases.

Finally, for Situation C, we must consider that the network for our company has expanded. This expansion can be found in Figure 7 of the appendix. With this expanded network, we must take into account that more files need to be transferred each day. The length of times for the new transfers are shown in Table 2 of the appendix. Along with more departments being added, some of our computers can also transfer more than one file at a time. These transfer capacities are represented in Table 3 of the appendix. Our goal is to find the shortest schedule for transferring files within this new network.

Assumptions:

For all situations:

1. When transferring files, both departments are being utilized. This is due to the fact that both departments are having files transferred. One must be open to receive the file while the other is in the process of sending the file.
2. The edges in the graph represent the transfer that must occur daily.
3. You can not pause a transfer and continue it at a later time
4. We can are going to assume that the time it takes to transfer is in minutes

Specific to Situation A:

1. All times to transfer files is 1, this is given in the problem.
2. The time to transfer each file is the same for each department

3. All computers can only transfer 1 file at a time. This indicates that one computer cannot be involved in more than one transfer.
4. 27 files must be transferred due to the network having 27 edges.
5. There is one transfer that occurs between each computer/department

Specific to Situation B:

1. We are following the same network given in Situation A.
2. The transfers with similar time frames should try to be grouped together in the same cycle. This is to ensure that a shorter transfer is not waiting for a significantly longer transfer.
3. Following assumptions three, four, and five from situation A.
4. The times to complete each transfer can be found in table 1.

Specific to Situation C:

1. Some departments can now be involved with more than one transfer at a time - see Table 3 for specific capacities.
2. Once one department is finished transferring its files, it may automatically be involved in a new transfer if needed. By allowing this we can make for a faster schedule.

Analysis of Models

Situation A

When working with the network given in Situation A, you can see the solution for the smallest cycle of time given in Figure 2. The different colors in this model represent different cycles (ex. Cycle 1=yellow, cycle 2=blue, cycle 3=red). There are multiple ways in which these cycles can occur. The yellow edges do not have to be associated with cycle 1, this model will still provide the shortest transfer times regardless of which colored cycle occurs first.

The important part of this model is that the colors(cycles) need to be separated in the way seen, no edges of the same color may be connected to the same department. This would interfere with the fact that a department may only be involved with one transfer at a time. Using the logic to solve for Figure 2, we establish that in a general case, the shortest possible number of cycles will be equivalent to the department with the largest number of edges(transfers). This also includes if a network involves departments connecting in a cycle. We built an algorithm that can determine an optimal solution that is detailed below and can be found implemented in python in our github repository (https://github.com/idugan100/communication-network):

Algorithm 1

```
Do for each degree of the graph:
    Sort nodes by the number of remaining edges to complete
    Using the order above do for each node:
        If the node is not currently “engaged”:
            Find an “uncompleted” edge on this node that goes to
            another “unengaged” node with the highest remaining
            degree

            If a eligible destination node is found:
                Set both nodes as “engaged”
                Set selected edge as “completed”
                Set edge color to corresponding color of the “round”
    Set all nodes back to “unengaged”
```

You can see the results of this algorithm when it is performed on that A situation in figure 2 and the results on another graph with additional nodes and edges in figure 3. Although we were unable to formally prove that this algorithm will yield the optimal scheduling of transfers for all cases, we tested over 10 different graphs and each time the algorithm found an optimal solution. A formal proof of this algorithm is a promising area for future work.

Due to the constraints and assumptions of situation A, the only changes that can be made are to add/remove a node/edge. The results of these operations are explored below:

	Add	Remove
Edge	This will never decrease the optimal time for all transfers but can increase it by 1 minute if this increases the degree of the entire graph	This will never increase the optimal time for all transfers, but can decrease it by 1 minute if this decreases the degree of the entire graph.
Node	This will never affect the optimal time for all transfer to occur because the newly added node will have no	Removing a node necessarily removes its edges. If removing this node decrease the degree

	edges/transfers to perform	of the graph, it will decrease the optimal time by the same amount that the degree was decreased

Based on the above chart we can perform some sensitivity analysis on our graph (figure 2). Adding an edge between nodes 14 and 24 will increase the optimal time by 1 minute because it increases the degree of the graph by 1. On the other hand, adding an edge between nodes 17 and 19 will not increase the optimal time because it does not increase the degree of the graph. Removing either of these edges will not result in a decrease in the optimal time because they do not cause a decrease in the degree of the graph. Adding a node (like node 29 in figure 3), does not affect the optimal time because it does not affect the degree of the graph.

These examples illustrate that our model is only sensitive to changes in the degree of the graph as long as the constraints (all transfers taking 1 minute, each department can only do one transfer at a time) are respected.

Situation B

For situation B we remove the constraint that all transfers take one minute and add transfer times from table 1. This additional constraint breaks the model we created in situation A. In situation A, because each transfer took the same amount of time, we could split the transfers into 3 separate rounds that each took 1 minute and produce an optimal solution. This will no longer work for situation B because we can not create 3 “rounds” of equal time. Our solution to this is to use a modified version of our algorithm from situation A to generate a dependency graph. A dependency graph is very useful for modeling situation B because it allows transfers to start as soon as both of the computers are free to perform it.

We can then use this dependency graph to find the critical path which will be the fastest time that all the files can be transferred while ensuring that no department is involved in two transfers at the same time.

The modification we made to our algorithm from situation A is to prioritize completing the longest edges so they can (hopefully) be completed in parallel. This also ensures that as much as possible, the longest transfers happen first so they don’t hold up the completion of all transfers. The modified algorithm is outlined below and the differences from Algorithm 1 are bolded. A python implementation of this can be found in b.py in our github repository (https://github.com/idugan100/communication-network):

Algorithm 2

```
Do for each degree of the graph:
    Sort nodes by the number of remaining edges to complete and then by
    the weight of the largest uncompleted edge from that node

    Using the order above do for each node:
        If the node is not currently “engaged”:
            Find an “uncompleted” edge on this node that goes to
            another “unengaged” node with the highest remaining
            degree. If there is a tie, chose the edge with the highest
            weight

            If a eligible destination node is found:
                Set both nodes as “engaged”
                Set selected edge as “completed”
                Set edge color to corresponding color of the “round”
    Set all nodes back to “unengaged”
```

The result of this algorithm can be seen in figure 4. The red round is first, blue is second, and yellow is third. We then use this ordering to generate a dependency graph of the transfers as seen in figure 5. The critical path of this dependency graph is outlined in green and shows that this particular schedule will take 23 minutes.

One of the difficulties of evaluating a model for situation B is that it is hard to tell if a solution is truly optimal. At first glance, it appears that the optimal solution for this graph would take only 21 minutes. If you look at node 20 in figure 4, you notice that it has three edges, each with a weight of 7. Because the computer can only perform one of these transfers at time, the optimal transfer time for this graph cannot be faster than 21 minutes (this creates a lower bound on the optimal time). This would seem to suggest that our model yields a suboptimal solution.

However, as we looked more closely at the graph, we struggled to find a schedule of transfers that would lead to a completion time of 21 minutes. Eventually we began to look for a proof that would show that a 21 minute transfer schedule was not possible.

Consider the sub graph of figure 4 shown in figure 6. If these transfers can be completed in 21 minutes, then each of the edges with length of 7 must be completed in consecutive time intervals shown as yellow, red, and blue. Each of the 9 minute edges must be completed during the time intervals when both its nodes are free. The time periods when each 9 minute edge can be completed are in parentheses. Because 9 is larger than 7, the 9 minute edges will need both time sections in parentheses to complete the transfer. Because transfers can not be paused and finished at a later time, we have reached a contradiction. There is no arrangement of the yellow, red and blue time periods that will make all the times in parentheses adjacent. This informal proof by contradiction shows that the optimal schedule for this graph cannot be faster than 23 minutes. Since we have a schedule that can perform all transfers in 23 minutes (figure 5), we have found an optimal solution. A formal proof of this would be a good item for future work.

For this particular graph, our technique of using Algorithm 2 to generate a dependency graph and then finding the critical path yields an optimal solution. However, we were unable to prove or disprove that this works in the general case. This is a promising area for future exploration.

One of the strengths of this model is that it yields a critical path for our schedule (green line figure 5). The critical path is the binding constraints of the model, so our model is very sensitive to changes in weight on these edges. Below is a table that explores how changes in weight to the three edges along our critical path while maintaining the weights of all other edges would affect the optimal solution.

Edge	Current Weight	Effect of Increase	Effect of Decrease
Node 14 - Node 18	9 minutes	Any increase of the time it takes to do this transfer will increase the optimal time by the same amount.	A decrease in this time will cause a decrease in the optimal time until this edge is below 7 minutes and is no longer a part of the critical path.
Node 14 - Node 20	7 minutes	Any increase of the time it takes to do this transfer will increase the optimal time by the same amount.	A decrease in this time will cause a decrease in the optimal time until this edge is below 3.2 minutes and is no longer on the

			critical path
Node 20 - Node 24	7 minutes	Any increase of the time it takes to do this transfer will increase the optimal time by the same amount.	A decrease in this time will cause a decrease in the optimal time until this edge is below 3.2 minutes and is no longer a part of the critical path

Situation C

In situation C we add some additional edges to the graph that can be seen in table 2. We also allow some of the computers to handle multiple transfers at once. The new transfer capacities are found in table 3. This gives us the graph shown in figure 7.

One way to find a lower bound on the optimal solution is to find the node with the highest degree and add the weights of those edges together. We can use this approach for Situation C. When considering the nodes, we see that department V_{24} has 5 edges connecting to it. Looking at table 3 in the appendix we also see that this node can only handle 1 transfer at a time. In other words, each of the edges connected to that vertex must occur at a different time. By using this logic, we know that the optimal solution can not be faster than the edges added together which is 29.6 minutes.

If this is a lower bound on the optimal solution, then if we create a model that takes that time, we know we have found an optimal solution. We were able to do this and our model can be seen in Figure 8. For our model, the edges labeled green will go first. The longest edge weight in this case is 9. We must note that in this cycle $V_{26}, V_{9}, V_{14}, V_{16},\ and\ V_{10}$ are all running more than one transfer at a time. These nodes can handle these transfers, as shown in Table 3 of the appendix. There are also some nodes that are shown to contain more than one transfer, when their transfer capacity is one. These nodes include $V_{6}\ and\ V_{5}$. The logic behind this is that the green round takes nine minutes, so the edge $e_{6}\ and\ e_{4}$ can go first and as soon as it finishes e_{5} will begin. This order can also be swapped, but $e_{6}\ and\ e_{4}$ should be going at the same time. By allowing this, the green cycle will stay within 9 minutes.

For the next round we will have the purple round. The longest length of time for this round is 7.1 minutes. Department V_{14} is the only department which is involved in two transfers at a time, but this node is capable of this. During the purple round, the edge e_{40} will start as soon as e_7 is finished. This will shave off some time on this transfer, making this round now 7 minutes long. We must also note that this transfer does not need to wait until the transfer e_{39} because department V_{10} can handle three transfers at a time.

The next cycle of transfers is shown by the yellow edges. The longest edge in this cycle is 6.1 minutes. We are able to make this edge bleed into the blue round, causing our longest edge to now be 4.2 minutes long. In other words, all of the yellow edges will run. The transfer of e_{42} will complete 4.2 minutes of its time during the yellow cycle and the last 1.9 minutes going at the same time as e_{34}. We also can clearly note that the final round, the blue round, will take 2.4 minutes to complete.

After making these clarifications, we can state that the green round is 9 minutes, the pink round is 7 minutes, orange is 7 minutes, yellow is 4.2 minutes, and blue is 2.4 minutes. This adds up to a total transfer time of 29.6, which is the optimal transfer time we found in the beginning. There may be other ways to accomplish this optimal transfer time, but this model is the way we discovered that worked.

When considering the sensitivity of this model, we can look at the department V_{24}. It currently has five edges and only the capacity to do one transfer at a time. If another transfer is needed to be done by this department and it still only has the capacity to do one transfer at a time, then the length of that edge will need to be added to our solution time. In other words, if it needs to do another transfer that takes 5 minutes long, then the optimal solution will change from 29.6 to 34.6. On the other hand, if we added an edge to V_{10} then the results would be different because that department has a capacity of three. We can also consider if we remove an edge from V_{24}. Our new binding constraint would be the department V_{20} because it has the capacity of one and has four transfers that need to occur. So, our new optimal solution time would change to 28.1.

Strengths and Weaknesses

Situation A

Strengths:

1. Found an optimal solution.
2. Able to prove that our solution is optimal
3. Model appears works for any general case
4. Model is flexible: cycles can go in different orders without affecting the result
5. Model is implemented entirety in Python making it easy to use on other graphs

Weaknesses:

1. Optimizes the finishing time, but not the mean finishing time
2. Unable to formally prove that the algorithm works in the general case

Situation B

Strengths:

1. Able to find an optimal solution
2. Able to prove that the solution we found was optimal
3. Algorithm + dependency graph technique is repeatable

Weaknesses:

1. Had to create a depency graph by hand which would take a long time for larger graphs
2. Unable to prove that our technique works in the general case

Situation C

Strengths:

1. Found an optimal solution
2. Able to prove it was an optimal solution

Weaknesses:

1. Non repeatable technique (trying different combinations of edges)
2. Does not solve the general case of this problem

Conclusion

Situation A

In conclusion, we have stated and proved that the shortest possible schedule for the network is three minutes long. Our model supports this and can be applied to the general case. Because we implemented our model in Python, it is very easy to generate an optimal schedule for any other graph. For future work involving this model, it may be important to consider transfers of priority. This may cause a longer amount of transfer cycles depending on where the edges are found. It would also allow us to provide more realistic models for companies that would fit their requirements.

Situation B

In conclusion, we were able to find the optimal solution of 23 minutes for that specific network. In our model analysis we proved why this model's optimal time would be 23 minutes rather than 21. Making our shortest possible time 23 minutes. Our specific model caused our solution to be slightly different from our initial thought, but we were able to find a method for the general case by creating an algorithm within Python. For future work we should consider if we are able to pause certain transfers in order to finish their times during different cycles. This may allow us to make our overall transfer time shorter than we initially found.

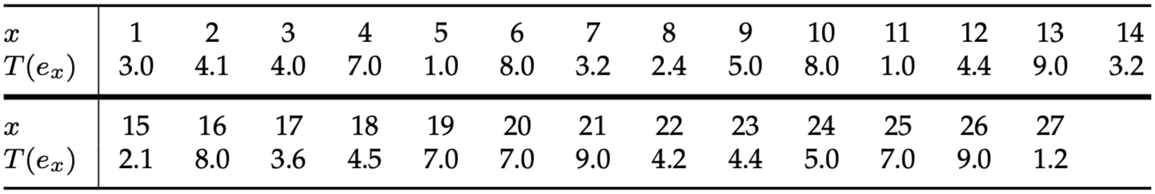
Situation C

In conclusion, we were able to discover our shortest possible scheduled transfer time would be 29.6 minutes long. In our model analysis we were able to prove that this was our shortest time due to the transfer capacity of V_{24} being one. After finding a model that matched this time, we accepted it as our optimal solution time. For future work, it would be important if we searched our model for other solutions that match our optimal time. This would allow us to find any flexibility within our network or changes that must occur if needed.

Appendix:

Table 1.

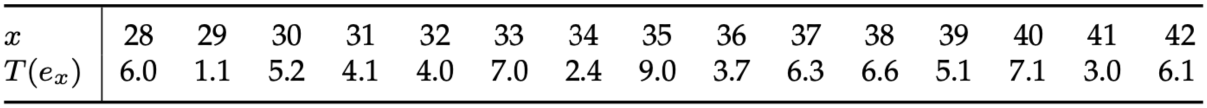
File transfer time data for situation B.



x	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$T(e_x)$	3.0	4.1	4.0	7.0	1.0	8.0	3.2	2.4	5.0	8.0	1.0	4.4	9.0	3.2
x	15	16	17	18	19	20	21	22	23	24	25	26	27	
$T(e_x)$	2.1	8.0	3.6	4.5	7.0	7.0	9.0	4.2	4.4	5.0	7.0	9.0	1.2	

Table 2.

File transfer time data for situation C, for the added transfers.



x	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
$T(e_x)$	6.0	1.1	5.2	4.1	4.0	7.0	2.4	9.0	3.7	6.3	6.6	5.1	7.1	3.0	6.1

Table 3.

Computer capacity data for situation C.

y	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$C(V_y)$	2	2	1	1	1	1	1	1	2	3	1	1	1	2
y	15	16	17	18	19	20	21	22	23	24	25	26	27	28
$C(V_y)$	1	2	1	1	1	1	1	2	1	1	1	2	1	1

Figure 1.

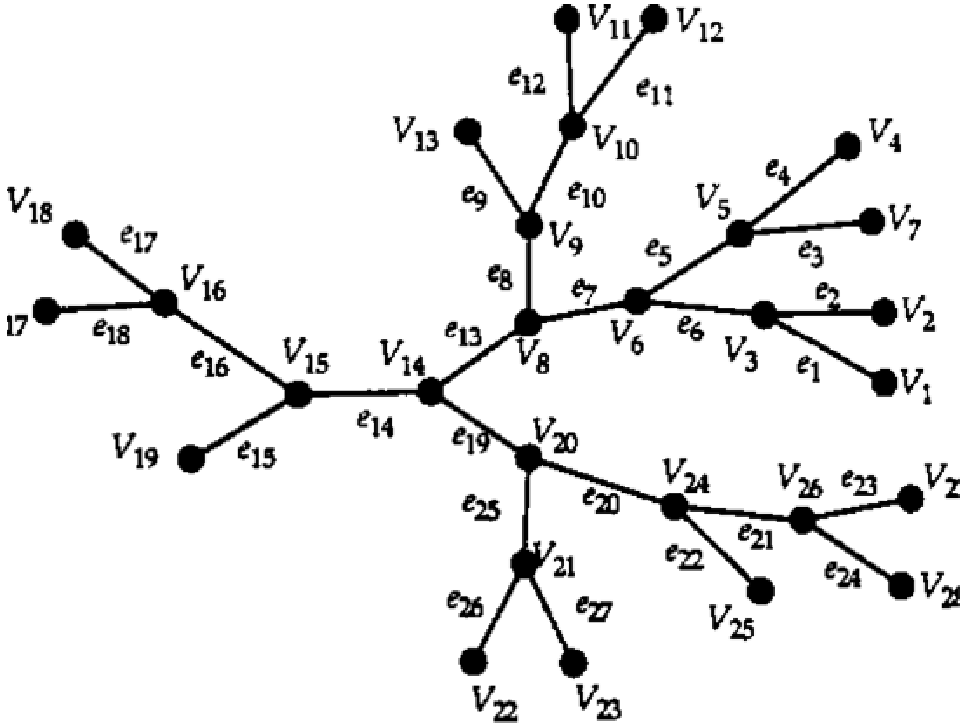


Figure 2.

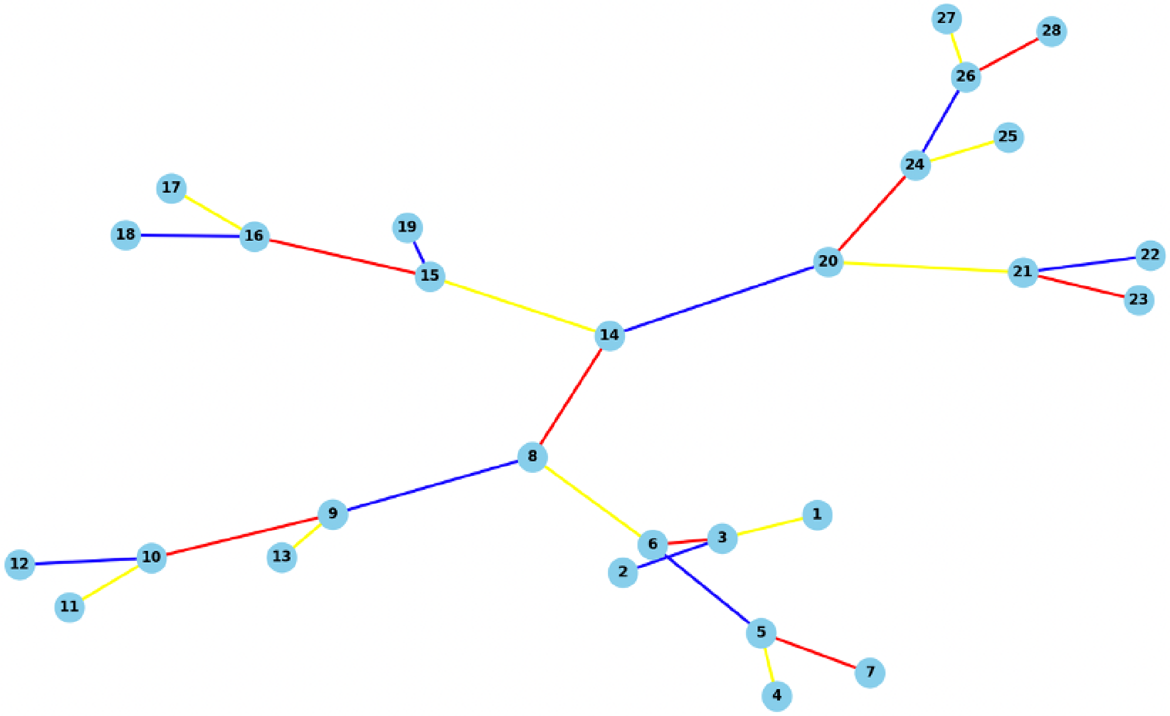


Figure 3.

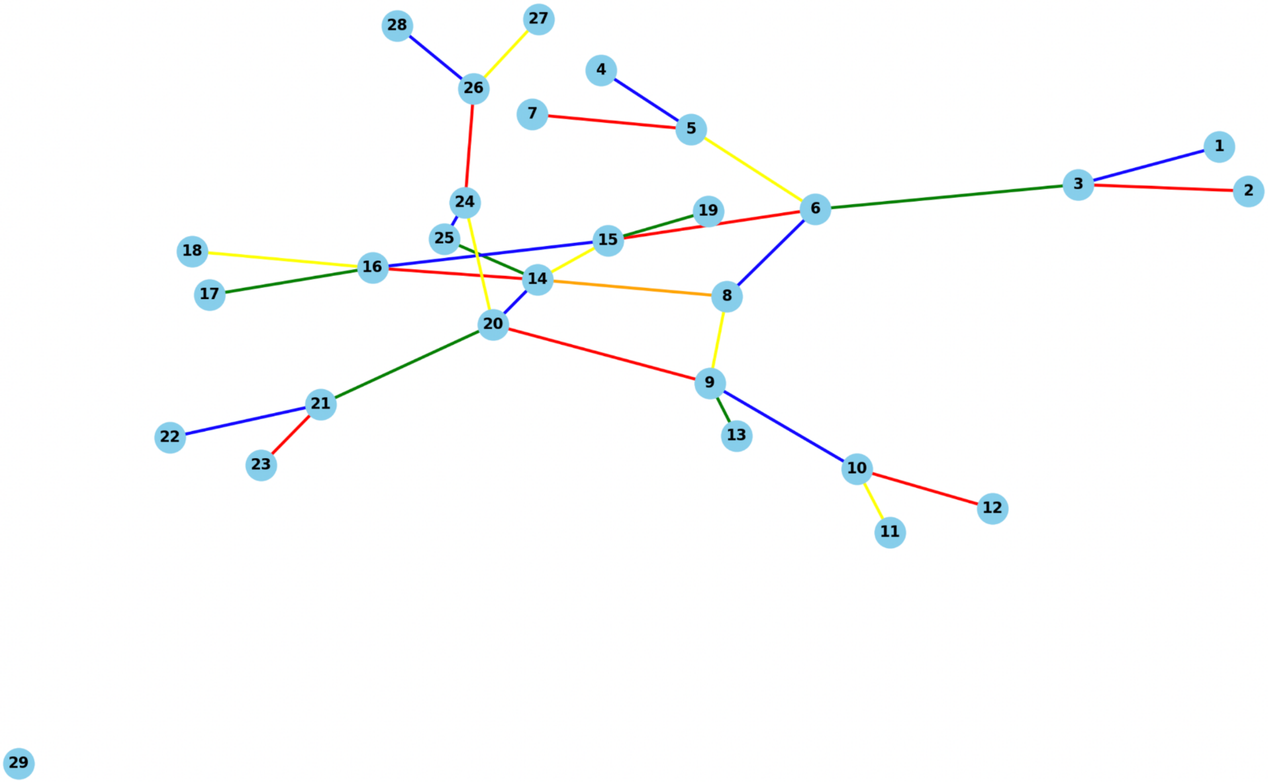


Figure 4

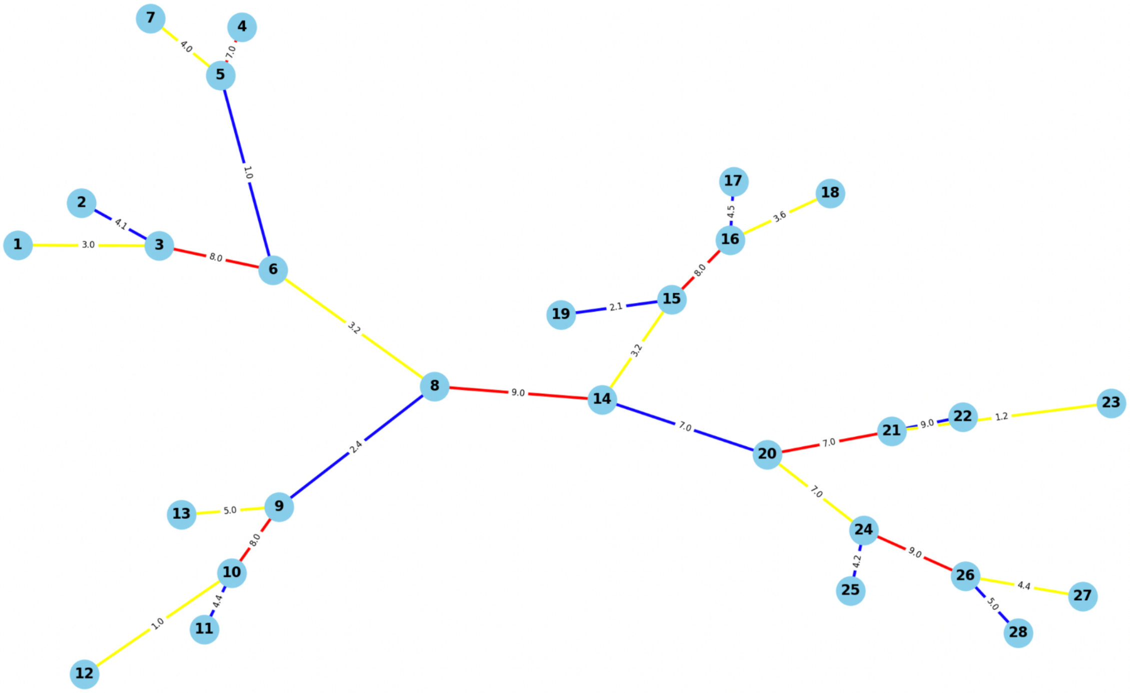


Figure 5.

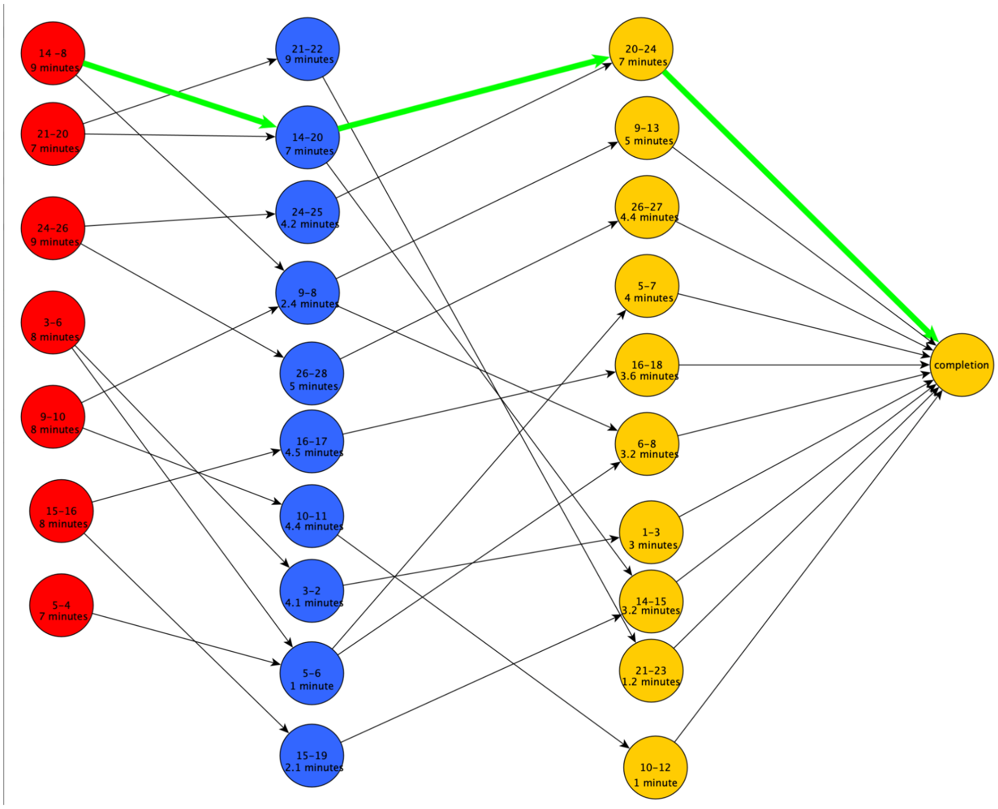


Figure 6.

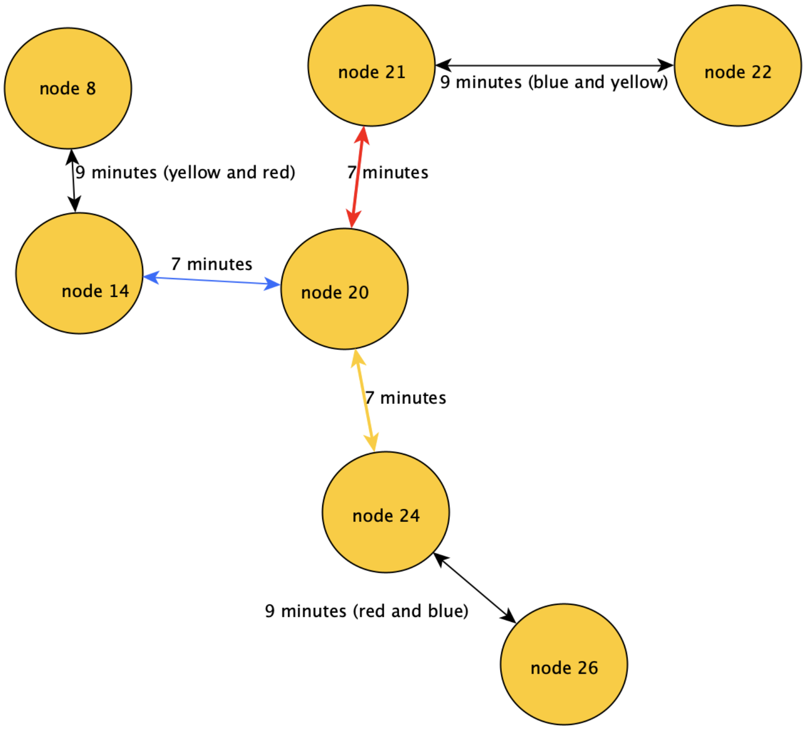


Figure 7.

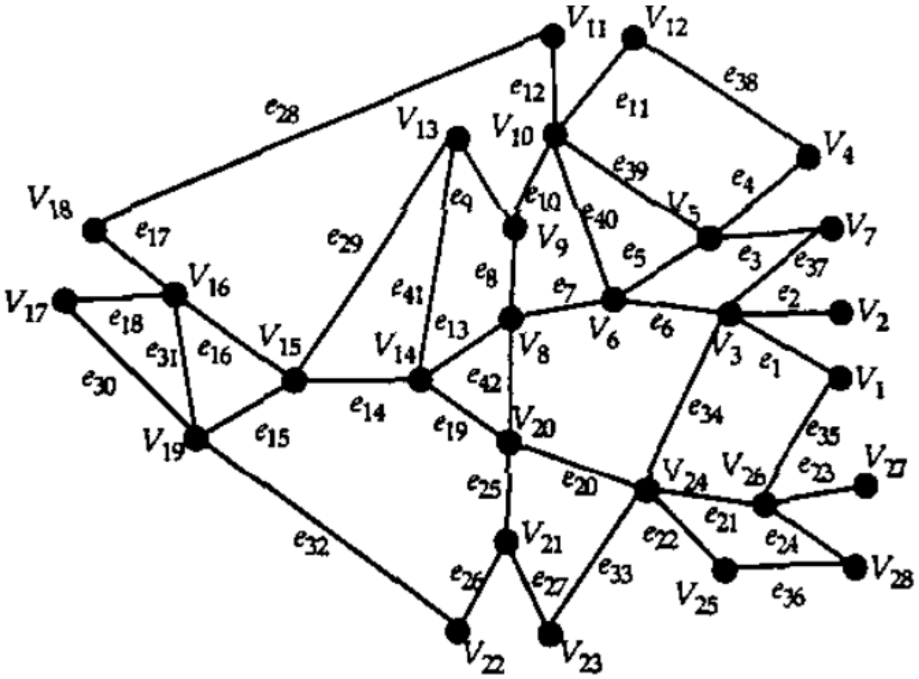
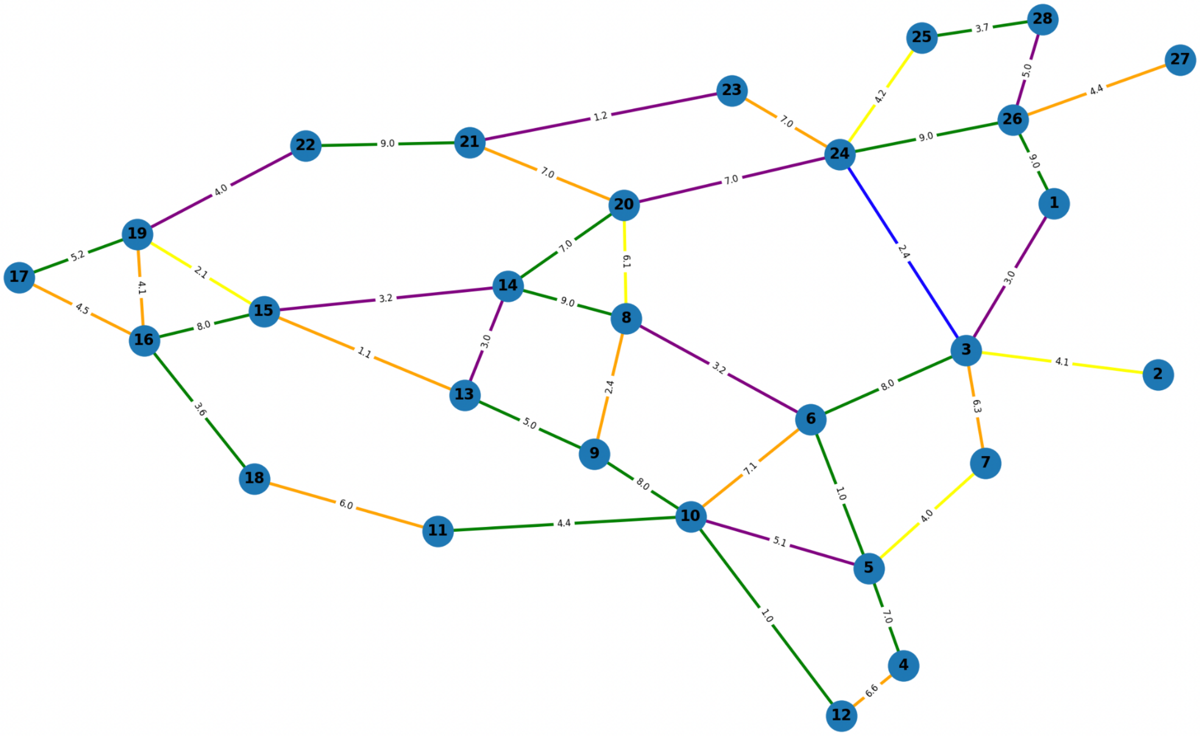


Figure 8.


28
3.7
25
27
5.0
4.4
23
4.2
26
7.0
1.2
9.0
22
9.0
21
24
9.0
7.0
7.0
20
1
4.0
19
7.0
2.4
5.2
2.1
6.1
17
4.1
14
3.0
3.2
9.0
15
8
4.5
8.0
16
1.1
3
3.0
3.2
2
4.1
8.0
13
2.4
6.3
6
3.6
5.0
7
9
7.1
18
1.0
8.0
4.0
6.0
10
11
4.4
5.1
5
1.0
7.0
4
6.6
12