Shared Shortest Paths in Graphs

By Brooks Emerick

Mentor: Dr. Sean McCulloch, Department of Mathematics and Computer Science at Ohio Wesleyan University

Terminology:

•A graph is a set of points called vertices that are connected by lines called edges.

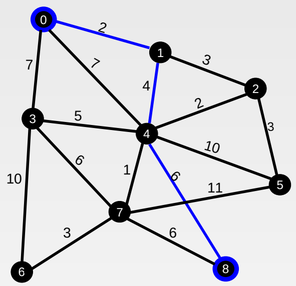
•A weighted graph is a graph that has an arbitrary cost assigned to each edge. The cost can represent anything from distance to the amount of resources needed to travel an edge.

•Graphs can be either directed or undirected. Directed graphs, or digraphs, have arrows as edges to specify the direction of motion; undirected graphs have lines as edges to represent two ways of motion.

•A journey is a designated starting vertex, the source, and ending vertex, the sink.

•A path is a list of vertices that describes a way to get from a source to a sink.

Example: An undirected, weighted graph with the shortest path (in blue) for a journey with source 0 and sink 8. Shortest Path: 0 – 1 – 4 – 8. Total Cost: 12.



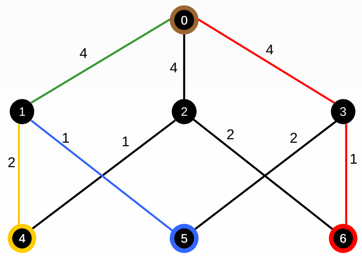
Problem:

We inspected the possibility of whether there exists an algorithm to produce the shared shortest path for an arbitrary number of journeys in any non-negative weighted graph. If two or more journeys have an edge in common, as described by their path, then they will share the cost of that edge. To share the cost means to divide the cost by the number of journeys are traveling on that edge. A journey has incentive to share with other journeys if its path can be shortened by making this decision to share. Therefore, to find the shared shortest path, every single journey or every group of journeys that are currently sharing must be the "happiest" they can possibly be with no incentive to relocate.

Equilibrium:

Equilibrium is the situation described above when every single journey or every group of journeys will not have incentive to move. This is the property of a graph that we must prevent in order to find a shared shortest path algorithm. However, there are cases in which no equilibrium prevails.

For example:



Sharing Edge (0, 1)
Costs:

$J_{0,4}$: 4

$J_{0,5}$: 3

$J_{0,6}$: 5

Sharing Edge (0, 2)
Costs:

$J_{0,4}$: 3

$J_{0,5}$: 5

$J_{0,6}$: 4

Sharing Edge (0, 3)
Costs:

$J_{0,4}$: 5

$J_{0,5}$: 4

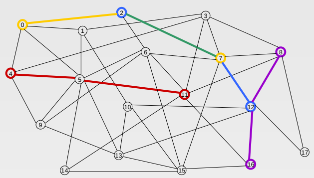
$J_{0,6}$: 3

NP-Complete:

Finding an algorithm to produce the shared shortest path with equilibrium is known to be NP-Complete. This means that there is no known way to solve the problem in polynomial time. In the shared shortest path algorithms that we created the problem was such that as the number of journeys increased, the time to perform the algorithm became exponential. Therefore, we examined the possibility of creating a heuristic that closely approximates the shared shortest path or finds non-equilibrium configurations in graphs, and runs in polynomial time.

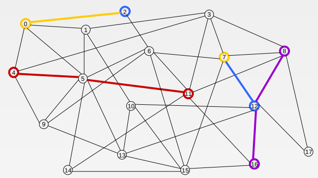
The Approximate Shared Shortest Path Heuristic:

Step 1:



•Using Dijkstra's Method, find the unshared shortest path from each journey's source to sink. If any paths are overlapping, compute the cost accordingly since they will now be sharing those overlapped edges.

Step 2:

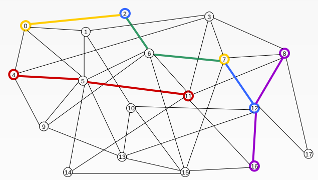


•Now delete the edge that has the largest coalition of journeys on it (2-7).

•The coalition consists of the journeys that are sharing the edge that is currently deleted.

•Leave the non-coalition members untouched during this deletion. A coalition member can share with a non-coalition member if it is beneficial.

Step 3:

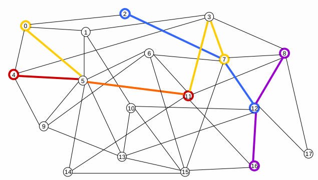


•Now repeat step one and find the new unshared shortest path of the coalition members only.

•If they overlap, compute the costs accordingly.

•If the new costs of at least one of the coalition members is shorter, then update from the previous graph to this new graph, and keep the total costs for every journey.

Step 4:



•Now repeat step 2 by deleting the edge that is most shared (2-6), and replace the edge that was deleted before.

•Find the new paths of the coalition members, given that they can share with a non-coalition member.

•If at least one member of the coalition gets shorter, update the graph and keep the costs.

•Repeat these steps until every desired edge is deleted.

Heuristics:

•By running different variations of our program, we found an approximate solution to the shared shortest path problem.

- •We changed the number of times to check each edge in order to find an edge that we may have missed earlier in the process.
- •Allowing the graph to be updated and the costs to be saved only if the majority of the coalition gets shorter rather than one member getting shorter. This parameter can also be tweaked so that the majority of the non-coalition members' costs get shorter.
- •We altered the required number of coalition members before the edge was deleted. If this parameter was set to zero, then every edge would be checked and deleted.
- •After every deletion, we can find the new paths of the coalition members without them knowing the position of the non-coalition members. Then, find the non-coalition members new path given the coalition members' new paths.

•By analyzing the graph every time there is an update in costs, we can find the most advantageous shared shortest path.

- •The minimum sum of all journeys' costs was considered an acceptable solution.
- •The ratio of cost-lost to cost-saved gave us a way to determine which edge deletion created the most savings compared to the graph before the deletion.

Results:

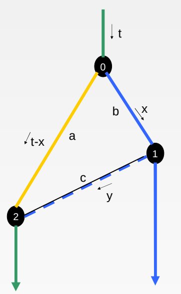
•Using the parameter that allows us to change the number of required members in the coalition before deletion, we found that, on average, 90% of these updated graphs had journeys that saved at least 5% of costs. Also, 60% of all updated graphs had journeys that saved at least 20% of their costs.

•In the heuristic, those 90% of graphs whose costs improve, on average, save at least 8% of their starting Dijkstra's shared cost. Also, 75% of graphs will have journeys that save at least 15% of this starting cost.

Other Interesting Results:

•We noticed that if a group of journeys are sharing and then split into two separate groups, then the two groups will never meet up again.

•Explanation: For the statement to be true, part or all of group *x* will leave the group *x* and will meet back with group *t-x*. Therefore, the entire group *t* will part ways at vertex *0*, group *x* will travel to vertex *1* while a group *y* will leave and meet up at vertex *2* with group *t-x*. (picture at right)



•Let the weights be *a*, *b*, and *c* as pictured on the graph. The smaller arrows show which groups are traveling along the indicated edges.

•Since we are assuming the statement is true, we have

$$\frac{a}{t-x} < \frac{b}{t} + \frac{c}{t-x+y}$$

for the group *t-x*. This inequality must hold for the *t-x* group to want to travel along the *a* weighted edge. Also, for the case of the group *x* and group *y*, we have

$$\frac{b}{x} + \frac{c}{y} < \frac{a}{t-x+y}$$

If we solve for *a* in both equations, we get

$$a < (t-x)\left[\frac{b(t-x+y)+c(t)}{t(t-x+y)}\right], a > (t-x+y)\left[\frac{by+cx}{xy}\right].$$

Clearing denominators and subtracting to one side yields

$$0 < (xy)(t-x)\left[b(t-x+y)+c(t)\right] - t(t-x+y)^2\left[by+cx\right].$$

•Now substituting any value between *0* and *t* in for *x* and likewise for *y*, with *y* < *x*, leads to a contradiction since we are only dealing with positive weights.

Conclusions and Future Work:

As of now, we have a heuristic that runs in $O(V^2E^2J)$ time that can produce a shared shortest path given any graph with any amount of designated journeys. We would like to be able to use more accurate heuristics to create better approximations. To do this we will have to run the program on a sample of graphs to gain statistics of how well the heuristics approximate the true solution. Creating different variations to the algorithm can only help get closer approximations. We would also like to gain more knowledge into finding non-equilibrium configurations. With this information, we could avoid non-equilibrium situations or describe more types of graphs that admit no equilibrium.

Acknowledgements:

•Dr. Sean McCulloch for his mentoring and help.

•The Physics and Mathematics/Computer Science faculty that were involved in the REU program for providing free food and valuable information at the Friday lectures.

•The Ohio Wesleyan University for making this research experience possible and for providing facilities and resources for me to use.