Different Optimal Solutions in Shared Path Graphs

Kira Goldner
Oberlin College
Oberlin, OH 44074
(610) 324-3931
ksgoldner@gmail.com

Sean McCulloch
Ohio Wesleyan University
Delaware, OH 43015
(740) 368-3663
stmccull@owu.edu

ABSTRACT
We examine an expansion upon the basic shortest path in graphs problem. We define *journeys* to be source-destination pairs in weighted and connected graphs, and allow them to equally split the cost of shared edges. In this new problem, there are multiple possible definitions of optimality. We investigate three: minimizing the total resources—the sum of the journeys' costs—of a graph's journeys; minimizing individual journeys' costs using analysis from game theory with an aim of stable formations called Strong Nash Equilibria; and minimizing the maximum cost that any journey in a graph has to pay, a cooperative solution. We developed heuristics that, given any weighted, connected graph and a set of journeys, can manipulate the journeys into routes that approach these definitions of optimal. Two versions, speedy and exhaustive, were developed of the Strong Nash Equilibrium heuristic. Results showed that the speedy version was equally as effective as the exhaustive version 99.0% of the time. 18% of the tests on the cooperative heuristic gave different results from different initial conditions, indicating potential room for improvement.

1. INTRODUCTION

The problem of shared shortest paths in graphs concerns any given weighted, connected graph with any number of specified journeys. We define a weighted graph as $G = (V, E)$, where V is a set of vertices, and E is a set of edges, connecting pairs of vertices together. Each edge is given a weight, which can be viewed as a "cost" to traverse that edge. We define journeys as pairs of a source vertex and a destination vertex. A path is a sequence of edges from a journey's source to its destination, and a journey's cost is the sum of its share of the costs of the edges in its path.

The well-known basic problem concerns finding the shortest paths in graphs given any set of journeys and a weighted, connected graph. Dijkstra's Algorithm [2] successfully finds the lowest cost path for each journey.

Our problem is an expansion of this. We permit journeys to equally split the cost of any edges shared amongst those using the edge instead of each individually paying the full price. In the situations we consider, any edge can be shared by any number of journeys, without restriction. This sharing condition gives journeys incentives to take paths that they might not otherwise take, depending on the paths that other journeys are taking and how they might be able to overlap.

Figure 1 is an example of this. While the sequence of A-E-C-D would cost 10 for an unshared path, because AE is shared by two journeys, each only pay 3.5 to traverse the edge. The shared cost of the A-E-C-D sequence is 5 as compared to the original route of the $A \rightarrow D$ journey, costing 7.

Additionally, allowing journeys to share the cost of edges has an effect upon optimality. We define the total resources of the journeys' current paths to be the sum of the costs of each journey on a graph.

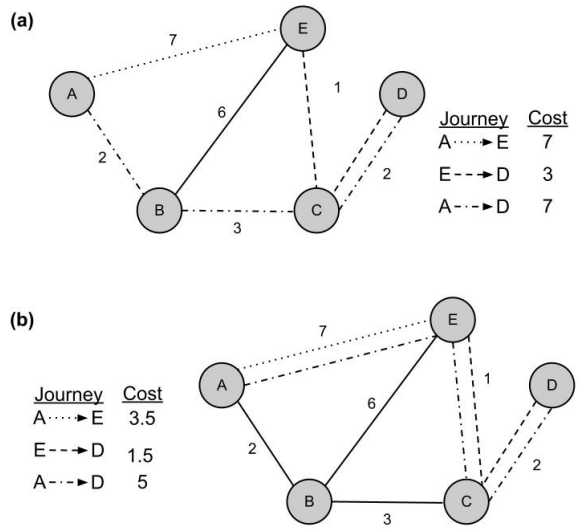

Figure 1. Graph (a) shows the three journeys routed by their traditional shortest paths. Graph (b) allows sharing and gives journey $A \rightarrow D$ incentive to use a different path.

In an unshared graph, finding each journey's own shortest path also minimizes the total resources used. However, in a shared path graph, to minimize total resources, journeys may take a slightly longer path in terms of shared cost. Such journeys may be sharing edges that largely reduce the shared costs of multiple other journeys, hence minimizing the total resources. This concept can be considered a global view—a perspective of the journeys considered as a whole.

For a local view—one where we look into each journey individually instead of grouping them together collectively—we can also see journeys as independent and inherently selfish, aiming to minimize the cost of their own paths with no regard to total resources. We can apply game theory to this concept for further analysis.

Due to these complications, optimality conditions must be specified when trying to find optimal solutions for the journeys of a shared path graph. This paper looks into a few different optimality conditions and how we have developed heuristics to approach them.

2. BACKGROUND

2.1 Past Heuristics

Past work [3] has been completed in developing heuristics that approach optimal solutions in minimizing the total resources of a graph's journeys. This problem is an instance of the Steiner Forest problem, a known NP-Complete problem [5], so heuristics were needed to approximate an optimal solution. Two heuristics specifically were quite effective and impacted our own research.

The first heuristic is the DEASE heuristic—Delete Edge And Share Edge. The heuristic assumes an initial routing of journeys created by some pre-processing step. (We use the unshared shortest paths connecting each journey, but any initial routing of journeys will do.) First, the heuristic considers every edge in the graph. If an edge has at least two journeys over it and is therefore a shared edge, then the edge is momentarily deleted. Any journeys that previously used the deleted edge are then rerouted by their new shortest shared paths. If enough journeys improve, then the journeys keep the alternative routes and the edge is replaced. If not, the edge and journeys are put back into their previous places.

By deleting shared edges and rerouting the journeys that use them, the DEASE heuristic encourages sharing, often reducing the costs of the total resources, although not necessarily of all individual journeys.

The second heuristic is the Spanning Tree Approximation. The heuristic uses Prim's Algorithm to find the minimum spanning tree of the graph in question. The journeys are then routed over the spanning tree. As each journey only has one possible path over the spanning tree, it is clear that sharing edges is encouraged. The paths over the spanning tree are then put back onto the full graph with all of its edges. The minimum spanning tree forces journeys to take not necessarily their

shortest paths, but sharing generally reduces the cost of total resources used by journeys in the graph.

2.2 Game Theory

In some heuristics, we apply game theory to shared path graphs. Journeys are assumed to be inherently selfish, looking for their shortest possible path with no respect to total resources or the costs of other journeys. Under these conditions, if a journey, given how all of the other journeys are presently routed, can change its strategy to give itself a lower cost path, then it will.

A Nash Equilibrium [4] is a situation where no journey has any incentive to change its strategy given how the rest of the journeys are behaving. Because no individual journey has incentive to move on its own, this is a form of stability. Nash Equilibria finding algorithms were already developed in past research on the subject [3].

However, the concept of Nash Equilibrium does not address the idea of multiple journeys strategizing together. Given how the rest of the journeys are behaving, a coalition may form and change their routes if every member of this group could improve by together choosing alternative paths. A Strong Nash Equilibrium [1] exists when no coalition of the journeys on the graph can together change their paths to benefit every member. In this case, journeys have absolutely no incentive to change their paths on their own or in a group. These formations are extremely stable and are the ideal formation for which we look.

If we refer to the three journeys present in figure 2 by their destination vertex—*E*, *F*, and *G*—we can see how the three possible coalitions cause the graph to defect. In graph (a), journeys *E* and *G* could both benefit by forming a coalition and taking alternate paths. In graph (b), the formation caused by this change, the coalition of *F* and *E* can defect to benefit both journeys. Graph (c) is the depiction of this, but the coalition of *F* and *G* will defect back to graph (a). Because the sequence of coalitions forms a cycle of defections, there is no Strong Nash Equilibrium in this graph.

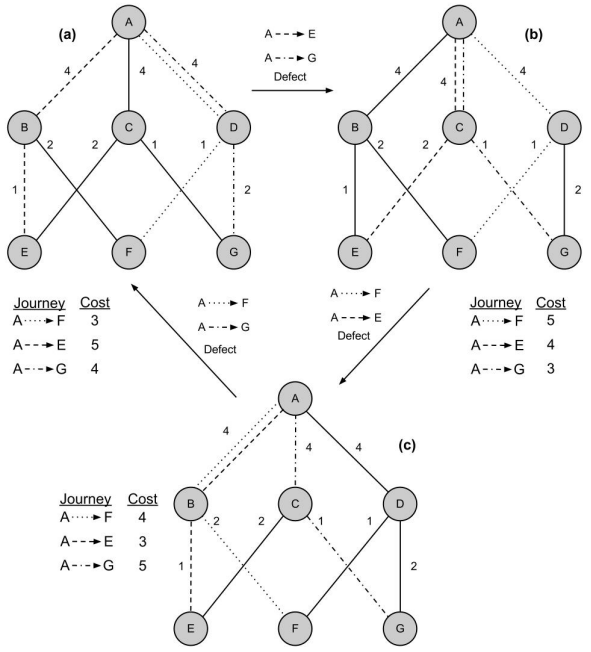

Figure 2. In this graph, coalitions will always have incentive to change paths. The graph will cycle through the above defections. No Strong Nash Equilibrium exists in this graph.

Nash Equilibria have been proven to exist in every graph [6]. However, figure 2 provides a counterexample that Strong Nash Equilibria do not exist in every graph. Furthermore, determining the existence of Strong Nash Equilibria (SNE) is $\sum_2^p$−Complete [7].

3. DEFINITIONS OF OPTIMALITY

In our work, we have considered three conditions of optimality for journeys in graphs. The first definition of optimality, investigated in previous work [3] is minimizing the total resources, or as previously discussed, minimizing the sum of the cost of all of the journeys in a graph. Our second definition is finding the lowest possible cost paths for each individual journey using game theory. Within this definition, we look for a Strong Nash Equilibrium, the most stable solutions where each individual journey has reached its

optimal path. The final definition pertains to minimizing the maximum cost that any individual journey in the graph must pay, and may be thought of as the cooperative or fair solution.

These different definitions of optimality have various real world applications. Minimizing the total resources used by journeys in a graph is very practical when the same agency is paying for the costs of all of the journeys, such as in a network of roads or telephone wires. It can be worthwhile to lengthen some journeys to minimize the total amount of concrete paid for by the government overall.

The game theory aspect of looking for Strong Nash Equilibria is applicable to situations like carpooling, where individuals can reduce their gas costs by splitting legs of journeys. In situations where Strong Nash Equilibria exist, because the solutions are very stable, highways or public transit systems could be designed off of the predicted stable behavior of individuals.

An application of a cooperative solution is considering a game theory-like example where journeys are inherently selfish, but an agency, such as the government, steps in and imposes regulations on the journeys to reduce the amount paid by outlying journeys.

4. STABILITY AND STRONG NASH EQUILIBRIA

4.1 SNE Heuristic

The Strong Nash Equilibrium (SNE) Heuristic determines whether or not Strong Nash Equilibria exist within a given graph, and finds one if they do. The heuristic uses graphs already routed by pre-existing heuristics. To attempt to find a Strong Nash Equilibrium, the heuristic would be given a maximum coalition of size of n-1 where n is the number of journeys. We call this the exhaustive version.

Pseudocode for the heuristic is as follows:

```
for each possible coalition size up to the
given maximum do
    try all possible coalitions of that size (in
    some arbitrary order)
        remove their current routings from the
        graph
        reroute them using shortest path and
        spanning tree heuristics
        if any re-routing exists where all
        coalition members improve:
            commit this new routing
            remember that we have changed a
            coalition's paths
repeat the above loop a number of times
equal to the number of journeys.
```

When the heuristic finishes, there are three possible outcomes. First, if no coalitions have ever changed paths, then the original graph is assumed to be in an SNE, at least for coalitions up to the given maximum size. The second outcome occurs if some coalitions changed paths, but none changed for at least the last iteration of the "try possible coalitions" loop. We assume that since the graph has stopped defecting that the graph has moved to an SNE. The final outcome is when coalitions have changed paths within the last iteration of the "try possible coalitions" loop. In this case, we conclude that that graph is caught in a never-ending cycle of defections. No SNE can be found for the graph in this instance.

4.2 Heuristic Effectiveness

The SNE Heuristic relies upon other heuristics. As it uses heuristics to determine if each coalition has a better rerouting that benefits every coalition member, it may be missing truly optimal routes. Additionally, it does not calculate the shared edge costs completely accurately when rerouting the coalition members via the Spanning Tree heuristic. Edge costs are estimated based on the number of non-coalition members using them, and these estimates are used when the minimum spanning tree is found. The inaccuracy in edge costs may lead to a

spanning tree other than the true minimum being used to reroute the coalition members.

A complication that we ran into in the early development of the exhaustive heuristic was that, in one case, the heuristic determined that a Strong Nash Equilibrium did not exist within the graph that actually had an SNE. The issue was that the exhaustive version was cycling through the same large coalitions and defecting with each of those. However, if it were to check the coalitions in a different order of size, it would defect in a different way and find a Strong Nash Equilibrium. We maintain a list of changed coalitions to keep track of the coalitions that cause the graph to defect. If the coalition has already caused change in the graph, the heuristic will postpone choosing that coalition and keep looking.

The exhaustive version of the heuristic performs closest to optimal when the given maximum coalition size is n-1. However, testing 2^n coalitions provides for an intractable running time. We looked into comparing the solution that the heuristic found when testing all 2^n coalitions with the solution that the heuristic found when only testing coalitions of size 2 and 3, pairs and trios, which we call the speedy version.

The results of our tests were extremely promising. 400 tests were run on a variety of vertex and journey ranges: the number of vertices in a graph ranged between 5, 10, 15, and 30, while the number of journeys given were either 5 or 10. Each test generated a random graph and random set of journeys with the given parameters: number of vertices, number of journeys, and a range for the edge costs. It then used four different pre-existing heuristics on the graph to generate four starting formations, and then ran the SNE heuristic on these formations. All of our tests are designed on graphs that are randomly created to be sparse, to encourage journeys that span several edges. Our sample graphs randomly create a connected graph with 1.2 times as many edges as vertices, with weights distributed over a range of 1 to 10. We have other graph generation algorithms that create graphs that encourage other kinds of sharing, and a possible avenue of future work would be to examine how our heuristics do on these graphs.

Table 1. SNE Heuristic Test Results

Vertices/ Journeys	% SNE found	% Speedy = exhaustive
5/5	95%	100%
5/10	90%	100%
10/5	98%	100%
10/10	93%	95%
15/5	95%	100%
15/10	88%	100%
30/5	100%	100%
30/10	94%	97%

Table 1 shows examples of the testing. The first two columns refer to the 8 different kinds of randomly generated graphs. Statistics for each are averaged over the varying graphs produced and the four starting formations for each on which the SNE heuristic ran. The "percent SNE found" refers to the how often the SNE heuristic determined the graph to either already be in or converge to a SNE. The last column reports how frequently the speedy version and the exhaustive version of the heuristic found the same routing of journeys in the graph.

Of the 100 different graphs, each with four different starting formations, the discrepancy was minimal between the solutions that the SNE heuristic found with the exhaustive version verses the speedy version. Only 4 of the 400 different graph formations did not have both the speedy solution and the exhaustive solution produce the exact same graph. The graphs for these four varied in number of vertices, but all were of the 10-journey variety.

Of the 100 graphs, there were 15 in which the SNE heuristic determined that there was no Strong Nash Equilibrium from one starting formation yet identified one from a different starting formation. We attribute this disparity to the assumptions made based on the outcomes of the SNE heuristic described previously.

5. COOPERATION

5.1 Cooperative Heuristic

Our third definition of optimality concerns minimizing the maximum cost that any journey pays in a graph. This solution can be considered cooperative, because, for example, instead of one journey paying 5 and the other paying 9 as in graph (a) of Figure 3, they could be forced into a formation where they each pay 7 as in graph (b), improving the maximum work that any one journey would have to do and making the costs a little more "fair."

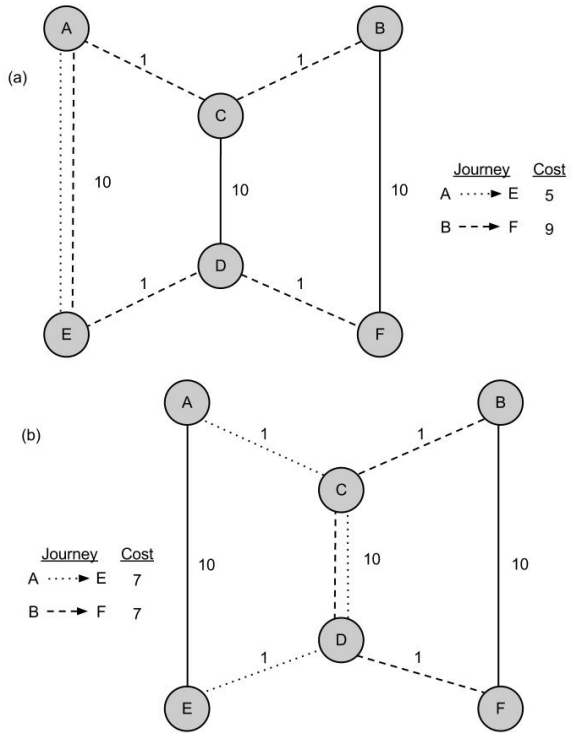

Figure 3. Graph (a) shows the standard routing of the graph with one journey paying more than the other, while (b) reduces the *maxJourney* to cost less with the same total cost for both journeys.

We define a graph's *maxJourney* to be the maximum of all of the graph's journeys' shared costs. Our goal is to minimize *maxJourney* so that the highest cost paid by any journey in the graph is decreasing, although individual journey costs may increase or decrease. The heuristic uses the same basic concept of the DEASE heuristic. When we delete a shared edge, the journeys that previously used that edge will be referred to as selected journeys.

Pseudocode for the cooperative heuristic is:

Given: a list of journeys $J_1 \ldots J_n$ already routed on some graph G

for each edge in G **do**

if more than one journey uses the edge:

save the selected journeys' current paths

delete the edge

reroute the selected journeys using the spanning tree heuristic, adding each journey back by its shortest path, or adding the journeys back by shortest path in reverse order

for each journey J_1 to J_n **do**

if J_i has been removed by deleting this edge: reroute it along its current shortest path

if the *maxJourney* of the graph improves or it stays the same but the total resources improve:

change the graph to use this formation

else:

for each journey J_n to J_1 **do**

if J_i has been removed by deleting this edge: reroute it along its current shortest path

if the *maxJourney* of the graph improves or it stays the same but the total resources improve:

change the graph to use this formation

else:

put the edge back and use the saved paths

When journeys are rerouted by adding them back by their shortest paths, they are added one at a time in the order that they are kept in the graph's list of journeys. This gives the journeys toward the end of the list an advantage. Journeys take the path of their shortest path visible at the time added back. Since more journeys are routed on the graph by the time the journeys toward the end of the list are added back, they can take into account all of the possible reduced edges, as opposed to the journeys at the beginning of the list who cannot take the later journeys into account. Journeys are rerouted in forward and backward order in attempt to minimize the effect that the order of rerouting has on the graph.

5.2 Results

Testing of the cooperative heuristic was extremely similar to that of the SNE heuristic. 100 graphs were generated with the aforementioned conditions, each graph having four different starting formations determined by pre-existing heuristics. These pre-existing heuristics were the aforementioned spanning tree approximation and DEASE heuristic in addition to a shortest path and traversal heuristic. The cooperative heuristic was then run on each of these starting formations. Data was then collected from the generated solutions: the cost of *maxJourney*, the total resources, and the number of defections. The average of these three measurements correspond to the last three columns of Table 2. The first two columns are the same as in the SNE heuristic tests.

Ideally, the heuristic should defect the graph until it finds the formation with the least possible *maxJourney* and least possible total resources with that *maxJourney*. However, of the 100 different graphs generated in the tests, 18 of them found different solutions from the different starting formations. For example, this happened in one of our test cases with 15 vertices and 5 journeys. The starting formation of the spanning tree approximation led to a cooperative solution with a *maxJourney* of 18, while the heuristic run on other starting formations of the same graph could only find a smallest *maxJourney* of 61. Since the heuristic did not find solutions as close to optimal on some starting formations as it did on others, the heuristic clearly has room for improvement.

Table 2. Cooperative Heuristic Test Results

Vertices/ Journeys	Avg Num Defections	Total Cost	Avg maxJourney
5/5	19.161	16.429	5.429
5/10	17.667	21.267	3.900
10/5	88.900	32.867	11.333
10/10	89.417	42.050	8.367
15/5	217.017	47.733	16.267
15/10	211.000	64.129	11.156
30/5	896.750	75.250	24.875
30/10	892.750	116.889	23.111

In looking at the number of defections from starting formation to solution, we were hoping to find a trend of which pre-existing heuristics find the closest-to-cooperative solution. However, there was an overall lack of trends observed.

An unexpected result was there was a correlation between 14 of the 15 graphs where the SNE heuristic found that there was no SNE from one formation and a different result for another and the graphs where the cooperative heuristic found differing solutions.

6. CONCLUSIONS AND FUTURE WORK

Thus far, we have successfully developed fairly quick and accurate heuristics that, given any weighted graph, and set of journeys, find fairly close to optimal solutions for several given definitions of optimal.

Future work on this topic might include a more effective and faster cooperative heuristic. We do not know currently how effective the heuristic is at providing the most cooperative solution because we have nothing to compare it to. We only know that the heuristic was effective in finding the best solution in small test graphs where the solution could be determined by eye.

Determining whether Strong Nash Equilibria exist in a graph is $\sum_2^p$-Complete [7]. However, perhaps by looking into graphs where SNEs exist and studying conditions consistently present, the question of existence could be answered much more quickly.

A field for future study could be differentiating between multiple Strong Nash Equilibria or Nash Equilibria and finding a way to assign values for best to worst formations. While journeys may not have any incentive to move, depending on how much sharing is required, this could impact this value for further assessment of stability.

Another question is to look into is the order of deletions in heuristics like DEASE and the cooperative heuristic. As mentioned in regards to the order in which paths are added back during steps of the cooperative heuristic, order makes a difference. Deletions cause the graph to start changing and can provide a different outcome than with a different order of deletions. Additionally, the order in which journeys or coalitions are considered in algorithms like the Nash Equilibrium finder and the SNE heuristic also provokes differing outcomes. Further study of the subject could prevent resulting graphs being biased by an arbitrary order selected.

7. ACKNOWLEDGMENTS

This project was funded by the National Science Foundation's Research Experience for Undergraduates Grant #1003992. Support was also provided by Ohio Wesleyan University.

8. REFERENCES

[1] Aumann, R. Acceptable points in general cooperative n-person games. *Contributions to the Theory of Games*, 4, 1959.

[2] Dijkstra, E. A Note on Two Problems in Connexion with Graphs. Numerische Mathematik, Vol. 1, 1959, pp. 269-271.

[3] Jagannatha, Z., Peterson, N., Quigley, S., Emerick, B., Earl, C., McCulloch, S. The Shortest Path Problem in Graphs. *MCURCSM 2011* http://personal.denison.edu/~lalla/MCURCSM2011/12.pdf

[4] Nash, J. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36(1):48-49, 1950.

[5] Ravi, R., Agrawal, A., Klein, P. When trees collide: An approximation algorithm for the generalized steiner tree problem on networks. Technical Report CS-90-32, Brown University, Providence, RI, 1990.

[6] Rosenthal, R. W. A class of games possessing pure-strategy nash equilibria. *International Journal of Game Theory*, 2(1):65-67, 1973.

[7] Scarcello, F., Gotlob, G., Greco, G. Pure nash equilibria: Hard and easy games. Journal of Artificial Intelligence Research, 24:195-220, 2005.