The Shared Shortest Path Problem in Graphs

Zeal Jagannatha
Ohio Wesleyan University
Delaware, OH 43015
zfjagann@owu.edu

Nicole Peterson
Ashland University
Ashland, OH 44805
npeterso@ashland.edu

Sean Quigley
University of Michigan
Ann Arbor MI 48109
seanpquig@gmail.com

Brooks Emerick
Shippensburg University
Shippensburg, PA 17257
emerick@math.udel.edu

Christopher Earl
Ohio Wesleyan University
Delaware, OH 43015
cwearl@owu.edu

Sean McCulloch
Ohio Wesleyan University
Delaware, OH 43015
stmccull@owu.edu

ABSTRACT

Our research into the Shared Shortest Path Problem (SSPP) attempts to route a number of paths, called journeys, in a graph where journeys share costs of common edges. There are several different ways to view the problem, measured by differing optimum solutions.

Our first type of solution attempts to minimize total cost, which is known to be NP-Complete. To satisfy this condition, we used an approach based on the Minimum Spanning Tree. Generally, this approach finds reasonably cost effective solutions, but more effective solutions can be found using other approaches. Our second metric is to treat each journey as an independent agent and attempt to find a Nash Equilibrium (NE) for the graph. From previous research on Congestion Games, we learned that NE always exist for our problem, and created an algorithm that finds them quickly, in practice. Additionally, while attempting to generalize Nash Equilibria, we investigated how Strong Nash Equilibria can also be applied to our problem. While this is one of the more effective optimality conditions, we found that it is likely NP-Complete, both to verify, and to find. In addition, we found that Strong Nash Equilibria do not always exist in graphs. As a result, we chose not to study this approach in detail, but have left it as a field for further study.

1. INTRODUCTION

1.1 Problem Definition

A well studied problem in graph theory is the question of the optimum way to find the path between two vertices in a weighted graph, which we call a *journey*. This journey is a pair of vertices which we seek to connect with some path. When the problem involves routing the journey such that the path connecting it has minimum cost, it is the well-known Shortest Path Problem. A generalization of this problem is to attempt to find the optimum way to route a group of paths through a graph where journeys share costs of common edges. This is the idea behind our problem: Given a graph G=(V,E) and a set of journeys $J \subseteq V^2$, we seek to find the "optimal" subset of E that connects all end points of journeys specified by elements J. To be more specific, a solution to the SSPP is a set $S \subseteq E$ such that $\forall(v, w) \in J$ there exists a path $P \in S$ such that P starts at v and ends at w and S satisfies some optimality condition.

This definition of our problem avoids defining optimality, so that we can have several different interpretations of it. This allows us to apply our solutions to a wider variety of real-life problems. We have found many ways of interpreting optimality, each with its own optimality condition and practical basis. One possible condition introduces a metric that minimizes total cost. Other conditions find solutions that are optimal for each journey or group of journeys, independent of other journeys. These methods for evaluating the problem often conflict, and must be approached separately, with different algorithms emphasizing different criteria. As a result, they give completely different results in most graphs. In practice, we have designed computational approaches for several different conditions that give results which emphasize the qualities of each condition, and outperform naive solutions based on the classic Shortest Path Problem.

1.2 About our Images

We use example images to aid the comprehension of concepts presented in the paper. To illustrate different journeys, varying dot styles are used for the edges which these journeys travel through. For edges that have no journeys over them, a solid line is used. For all other journeys, various (non-solid) dot styles are used. In the case that an edge has been deleted, the edge and its associated cost will be lightened, so that they will not appear as dark as other edges.

For convenience, labels are placed beside each image showing what dot style each journey has and what their cost is (if any). Optionally, this label may only be placed once for each figure, since the dot style remains constant throughout each figure.

1.3 Possible Optimality Conditions

When thinking about each of the optimality conditions, it is helpful to have a frame of reference with which to compare them. For this, we used the simple Shortest Path problem. We chose to use this as our basis because it is more intuitive to someone with little experience studying the SSPP, and also because it is easy to compute efficiently. Using this as the basis for comparison for our graphs allowed us to compare algorithms to so called 'basic' solutions, or solutions that were found using the Shortest Path approach.

The Shared Shortest Path Problem is defined as the problem of finding the optimal way to route a set of journeys through a weighted graph. As we mentioned above, the word "optimal" is not strictly defined, so there are many different criteria which we can apply to find solutions.

One possible optimality condition is simply the minimal cost metric, which seeks to minimize the total cost of the edges used by any journey in the entire graph, regardless of each journey's cost. This has applications in network design where there is a central authority overseeing the design of a network. For example, if we wanted to build a highway system that connected several major cities while minimizing the cost of materials, we could apply this approach to find a cost-effective solution to this situation. We would simply represent each of the people that want to travel between cities as a journey, and then route these in the current network of roads. The result of the algorithm would be some way of organizing the journeys so that they were minimal cost. The edges that are used by many journeys, that is, the roads that would be most heavily demanded by the population, should be the ones that highways are built along. This would allow us to build highways along these paths, reducing transportation costs along them, in addition to building costs for the highway.

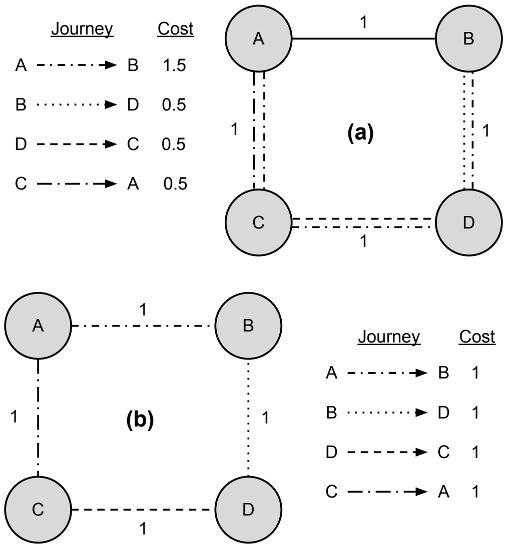


Figure 1: An example where the Nash Equilibrium is not minimal cost. (a) is one of the minimal cost solutions, but (b) is the stable Nash Equilibrium.

An alternate goal would be to find a configuration of journeys that attempts to find "stable" solutions from which journeys are unlikely to deviate. Solutions of this sort are graphs that have the property that no single journey can find a better path on its own, and are called Nash Equilibria [5]. The main application of this approach would be to try to design a network system so that independent agents in the graph are in stable configurations, and would be content with their paths. This is mainly used when there is no single authority responsible for finding minimal cost solutions, and the agents are allowed to act independently and, perhaps selfishly. As a result, solutions that attempt to approximate this method may not necessarily be minimal cost, but instead may pay a higher cost so that each agent is in a minimal cost path. Figure 1 illustrates the fact that these two solution methods are sometimes mutually exclusive.

Another way to attempt to find these solutions is through Strong Nash Equilibria. This is a much more complicated problem, as it has been shown that finding Strong Nash Equilibria in general is Σ_2^p-Complete [3]. In addition, there exist graphs that do not have Strong Nash Equilibria at all for our problem, as demonstrated below. For these reasons, we have done a preliminary investigation into Strong Nash Equilibria, but more work remains to be done.

2. RELATED WORK

A variety of work has been done that applies to the SSPP. Much of this work has been done in the field of Game Theory. One concept that is very closely tied in with the SSPP is that of Nash Equilibria [5]. Nash Equilibria are extremely useful when trying to analyze and find equilibrium-based solutions. A Nash Equilibrium is a configuration in which no journey can change its path to find a lower cost solution. It has been proven that NE always exist for our problems, using the concept of Congestion Games [7]. Congestion Games are games in which each player has a finite set of resources whose costs depend solely on the number of players using them. Rosenthal proved that this class of games always has Nash Equilibria by use of a potential function based on the resources and the number of players using them [7].

An even more rigorous form of Nash Equilibria is the Strong Nash Equilibria [2], abreviated as SNE, which refers to solutions in which no group of players can change their decisions to benefit every member of the group. While these solutions have usually a lower total cost than Nash Equilibria, Strong Nash Equilibria do not always exist. Figure 2 gives an example of a situation where coalitions of two journeys can cyclically defect leading to an infinite sequence of defections. Additionally, Strong Nash Equilibria are likely NP-Complete to find.

In the graph in Figure 2, any attempt to find a Strong Nash Equilibrium in the graph fails, since move than one journey can alter their path to find a new, lower cost path at any time. A walkthrough of the defections is given below.

In Graph (a), both journeys $A \to E$ and $A \to G$ can defect to find a better path, through edge (A, C), reducing their costs to 4 and 3, respectively. This gives us Graph (b) This leaves journey $A \to F$ alone and with an increased cost, giving it incentive to defect. With the help of $A \to E$, it does defect, so both $A \to E$ and $A \to F$ share edge (A, B). This

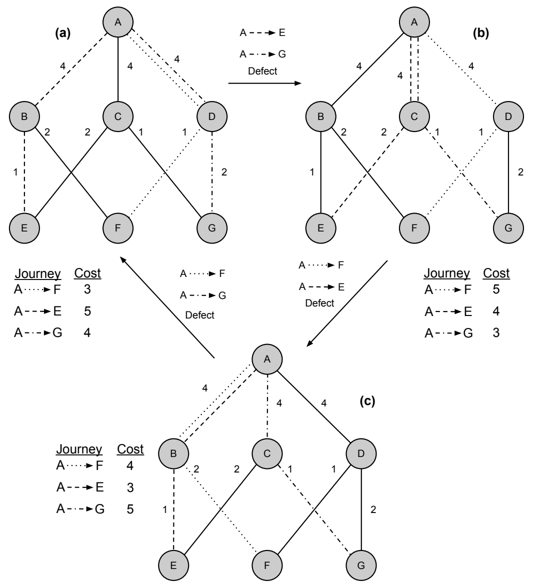


Figure 2: Any configuration of journeys in this graph allows us to defect two journeys at once to achieve a better cost for both. Thus, a Strong Nash Equilibria does not exist.

is depicted in Graph (c). Finally, journeys $A \to F$ and $A \to G$ can defect to reduce their costs once again, resulting in Graph (a). This circular loop of defections is sufficient proof that Strong Nash Equilibria do not exist in all graphs.

3. FINDING SOLUTIONS

Since there are a variety of ways to approach this problem, we attempted to apply computational methods to each approach to generate results emphasizing each of our optimality criteria. For each of the optimality conditions, we investigated several approaches to solutions, and evaluated them for quality and efficiency.

3.1 Spanning Tree

The main approach we took to the minimal cost criteria is the Spanning Tree Approximation, that lowers the maximum bound on the cost of a solution by removing unnecessary edges from the graph. When implementing this solution, we were unable to give absolute minimal solutions in polynomial time since that problem reduces to the Steiner Tree Problem[1]. However, we were able to provide rea-

[1]Since solutions may be disjoint, finding minimal cost solutions actually reduces to finding Steiner Forests, a generalization of the Steiner Tree Problem which is also NP-Complete.[1]

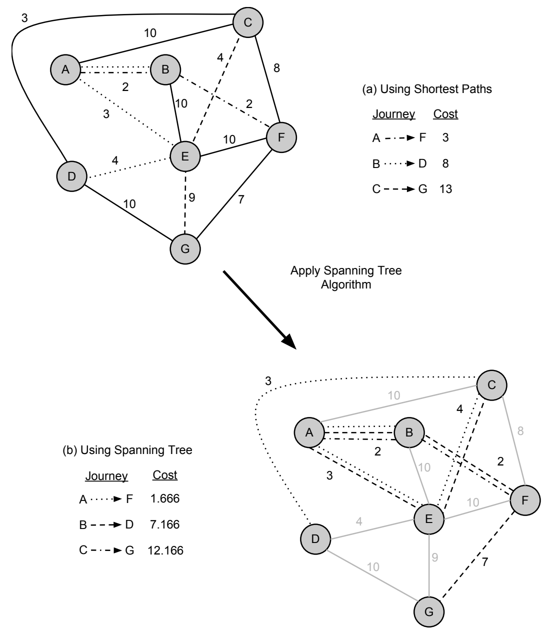


Figure 3: In this graph, the spanning tree allows us a quick way of reducing the cost of a solution.

sonable solutions that do significantly better than basic solutions and do so in a very short amount of time[2]. Given a particular weighted graph G and set of journeys J, the Spanning Tree Approximation finds the Minimum Spanning Tree of G and then routes the journeys in J on this Spanning Tree, forcing more journeys to share. Figure 3 shows an example of this. Notice how giving each journey just one way to connect forces sharing along common edges (AB, AE, BF, CE), leading to a reduction in each journey's cost relative to the basic Shortest Path solution. In general, it has been shown that this approximation will give a maximum total cost of twice the optimal solution's cost [4].

The motivation behind our Spanning Tree heuristic is the realization that the total cost of a solution is just the sum of all edges used by any journey. Our heuristic tries to minimize that sum by using the Minimum Spanning Tree of the graph. Because each journey in a Spanning Tree has only one possible path to connect its endpoints, the computational complexity of the approach is greatly reduced, and journeys were forced into sharing with other journeys, since there were significantly fewer paths which each journey had the option to take.

[2]Both practically, and in Big-O terms.

In example below, we begin with a graph and find its Spanning Tree. Once this has been done, we reroute each journey in the new graph. Since this new graph has fewer edges than the original graph, we reduce the upper bound on the total shared cost. In this case, we reduce the cost of all of the journeys, however, this is not always the case. There are cases in which we can actually force a journey to travel along a less profitable path when we find the minimum spanning tree.

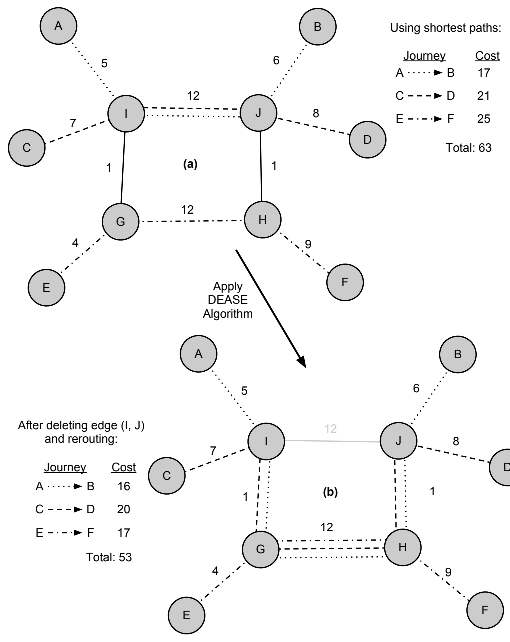
3.2 DEASE Algorithm

Another way to minimize costs is to attempt to encourage or force sharing between journeys. The DEASE (Delete Edge And Share Edge) algorithm does this by forcing large coalitions to form in the graph. It forms these coalitions by deleting edges that are used by smaller coalitions or individual paths. This forces the journeys to defect to different paths, forming larger coalitions, and reducing the total cost.

Pseudo-code for the algorithm is:

```
Route each journey initialy by finding its unshared shortest path.
repeat
    Mark any edges that are shared by more than one journey.
    Keep track of each journey's current shared cost.
    Delete the edge that has the most journeys on it.(This will result in any journey that used that edge to no longer have a path.)
    Reroute the affected journeys, using a standard unshared shortest path algorithms (such as Dijkstra's algorithm) taking into account the new shared costs of each edge.
    Compare the current costs of each journey to their old costs before the deletion
    Move any journeys that improved from their old position to their new one.
    Replace the deleted edge back in the graph.
until All shared edges have been deleted once.
```

In the example above, we can see how the algorithm forces coalitions of journeys to form. It does this by selecting an edge that is shared, in this case, the one in the top-center of the graph, and deletes it, rerouting any affected journeys. The journeys are rerouted and find a new path that is still shared. We then add back in the deleted edge, to reform the original graph, however, since the new path for both journeys is cheaper than their



Using shortest paths:

Journey	Cost
A ····► B	17
C - - -► D	21
E -·-·► F	25

Total: 63

After deleting edge (I, J) and rerouting:

Journey	Cost
A ····► B	16
C - - -► D	20
E -·-·► F	17

Total: 53

Figure 4: An example of DEASE algorithm.

former path, they remain on the new path rather than moving back to the old path.

The algorithm then selects another edge to delete, in this case, the edge directly below the previously deleted edge. Once this edge is deleted, the leftmost journey defects down to the journey below it, forcing the other affected journey to return to its original path. Since if the rightmost journey had defected first, the left one may have rejoined it, we can see that the order of the defections matters. In a similar way, the order of the deletions matters. We have left these concerns to futher study.

3.3 Nash Equilibria

If our optimality condition is to find a Nash Equilibrium, we need a different approach from the previous, although the Spanning Tree and DEASE heuristics may make good starting points. The Nash Equilibria Algorithm (NEA) will find Nash Equilibria in graphs by repeatedly asking each journey whether or not it can defect to a better solution until no journey can, and we have reached a Nash Equilibrium. With help from the field of Congestion Games, we have proven that this algorithm can always find a Nash Equilibrium in any graph. Unlike the Spanning Tree approximation, the NEA must be given an initial configuration of paths that connect each journey's end point and start point. This can be obtained through a number of ways,

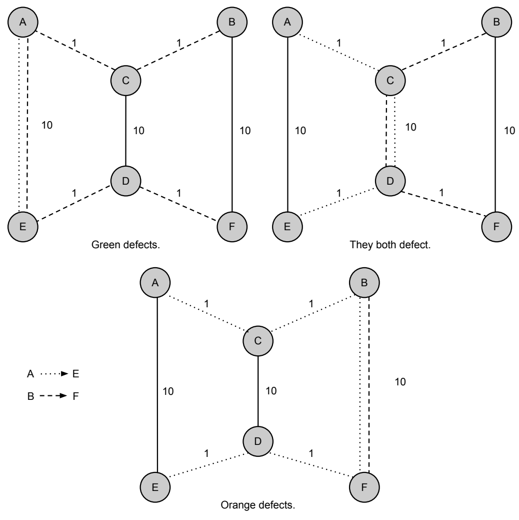


Figure 5: A graph in which there are more than one Nash Equilibria.

such as, finding their shortest single paths or finding a spanning tree for example. If the algorithm is given the same graph and a different initial configuration or a different order of deviations is chosen, two different Nash Equilibria can be found. This is true whenever more than one Nash Equilibria exist in a graph, which is a common property of most graphs. Figure 4 shows an example of this behavior.

The algorithm to perform this is quite simple:

```
Start:
for each journey J do
    Find the current shared cost of each edge
    Reroute J, using a standard Shortest Path algorithm on the graph with these updated edge costs.
    if J can reduce its cost by switching to a different path then
        Record the improved cost
    end if
end for
if any journey can reduce its cost by switching to another path then
    Defect the journey that has the largest change in cost
    Goto Start
else
    Since no journey can improve, we are in a Nash Equilibrium.
end if
```

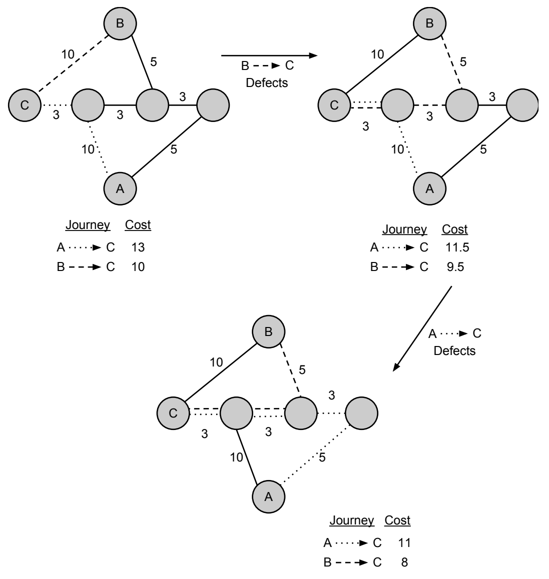


Figure 6: An example of the NEA. The starting configuration is given from the shortest paths for each journey.

Using a result in previous game theory work [7], we can prove that this algorithm always converges. This is done by providing a potential function of the solution the total cost and allowing each player to deviate and observing the change in potential. Since the potential always decreases, but is bounded below, it must converge.

4. FUTURE WORK

Although we have found an algorithm that always finds Nash Equilibria for our graphs, we have not been able to find either a bound on run-time or find bounds on the costs of configurations it produces. Both of these are goals for future work.

Strong Nash Equilibria are intuitively Σ_2^p-Complete for our problem since it is necessary to check each subset of journeys to even verify a particular configuration is a Strong Nash Equilibrium[3]. However, we have not proven that SNE are Σ_2^p-Complete to find or verify for our problem. In addition, we do not know if there are sufficient conditions for Strong Nash Equilibria to occur in graphs. If there is some sufficient condition, can it be found algorithmically? If so, what is the complexity of such an algorithm? These are all questions we hope to answer through future research.

Finally, we would like to broaden the scope of the SSPP. In our case, we only examined graphs for the SSPP, but it can also be applied to Euclidian Geometry to give us the Euclidian Steiner Tree Problem [6], which has further applications in network design.

[3]Of which there are $2^{|J|}$

5. ACKNOWLEDGEMENTS

Support for this work was provided in part by the National Science Foundations Research Experience for Undergraduates, grant numbers 0648751 and 1003992. Support was also provided by Ohio Wesleyan University's Summer Science Research Program.

In addition, we would like to thank Ronald Fenelus, whos valuable opinions provided us with much of the insight for this project.

6. REFERENCES

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

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

[3] F. Scarcello G. Gottlob, G. Greco. Pure nash equilibria: Hard and easy games. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.

[4] P. Plassmann M. Bern. The steiner problem with edge lengths 1 and 2. *Information Processing Letters*, 32(1):171–176, 1989.

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

[6] et al. P. Crescenzi. Minimum geometric steiner tree. *A Compendium of NP Optimization Problems*, 2000.

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