INTRODUCTION

Braess’s paradox is a situation that might occur when new edges are added to a network, where a newly created path degrades the transportation time. One might think that creating additional paths would improve transportation time; however, Braess’s paradox shows that the transportation speed can be degraded by adding certain paths because the drivers follow a game theoretic strategy without coordinating between each other.

A brief introduction regarding how network traffic is affected by game theoretic decisions and the occurrence of Braess’s paradox can be found in reference \(^1\). A proof of Braess’s Paradox is shown in large random graphs in reference \(^2\), while several guidelines on how to avoid Braess’s Paradox is presented in reference \(^3\). Braess’s ratio with respect to the maximum experienced latency of a flow particle is introduced in reference \(^4\), where the authors discuss how Braess’s paradox instance can be found if the Braess’s ratio is strictly greater than one.

In this paper we propose a method to find the edge causing Braess’s paradox, if any, in order to eliminate it. The proposed method is simulated and the results of the simulation are plotted and discussed. Braess’s paradox is usually seen in large and in over-connected networks; however, to simplify the discussion when applying our algorithm, we use a simple network from reference \(^1\) that is shown in Fig. 1. We consider two cases, directed and undirected graphs. In the directed case, the cars can only move in the direction of the arrows shown in Fig. 1. For example, they can move from 1 to 2, but not from 2 to 1, and so on. On the other hand, in the undirected case, the cars can move freely in any direction, for example both 1 to 2 and 2 to 1 are possible moves. The proposed method can be applied to larger networks in order to improve the transportation time by eliminating the edges causing Braess’s Paradox.

In our proposed method, we rely on simulation in order to find the route used when Nash Equilibrium is achieved. The route during Nash Equilibrium will be made of several edges and will cost the cars a certain transportation time. It

---

**Detecting Edges Causing Braess’s Paradox Via Simulation**

Toshiki Kojima*, Ahmad Bazzi** and Yoshikuni Onozato***

**Abstract** Adding edges to a network does not always improve the transfer/transport speed. Braess’s paradox is a situation that is caused by an added edge which degrades the overall transfer time instead of improving it. As a result, removing the Braess’s paradox causing edge will improve the overall transport time. In this paper, we propose an algorithm that pinpoints which edge is causing Braess’s paradox if it is occurring. Consequently it allows us to prevent Braess’s Paradox from occurring and to improve the total transportation time.

**Key words** Braess’s paradox, Game theory, Nash equilibrium, Transportation
is assumed that Braess’s Paradox is caused by a single edge out of these edges. Using simulation, the program deletes one edge of the n edges and finds the Nash Equilibrium of the new network and recalculates the transportation time. Then the program deletes a different edge of the original network and finds the new Nash Equilibrium through simulation and the new transportation time. We are searching for the one edge causing Braess’s Paradox; therefore, only a single edge is being deleted at a time and any previously deleted edges are restored. Roughgarden shows that the problem is hard to solve mathematically; however, we believe that it is more feasible to solve via simulation.

**Eliminating Braess’s Paradox**

In this section, we present our proposed method to optimize a traffic pattern. The steps are as follows:

1. First, check whether Braess’s paradox is occurring.
2. If it is occurring, create new networks by deleting one of the edges used in the traffic path.
3. Find the Nash equilibriums and the average travel time of each network.
4. Compare the time cost of each of the created networks with the time of the original network, and adopt the least expensive one.

The algorithm of the program is shown in Listing 1.

For example, consider the routes shown in Fig. 1 and suppose that 4000 cars want to get from city 1 to city 4. Each edge is labeled by the time cost it incurs the travelers. The edge between vertices 2 and 3 has zero cost, while the edges between 1 and 3 and between 2 and 4 have a time cost of L. To simplify our calculation, let L = 45. The edges between 1 and 2 and between 3 and 4 have a variable time cost directly proportional to the number of passing cars.

Each driver can choose among the following 3 routes to get from city 1 to city 4:

1. The path 1 -> 2 -> 4
2. The path 1 -> 2 -> 3 -> 4
3. The path 1 -> 3 -> 4

It is important to note that the total travel time (cost) of these different paths depend on the choices made by the other drivers. We assume that the choice of the drivers follows a game theoretic approach. This is because each driver has an incentive to decrease his total travel time regardless of the effect on the other drivers. Nash equilibrium is achieved when no driver has an incentive to change his path, in other words, no driver can decrease his total travel time. The Nash equilibrium of this network is established when all the cars use the route through 2 and 3 (the path 1 -> 2 -> 3 -> 4) resulting in a travel time of 80 minutes.

There are several other scenarios that the drivers could have chosen, but they are not Nash equilibriums. For example:

1. All the drivers can take the path 1 -> 2 -> 4. In this case the cost of the edge 1 – 2 will be x/100 = 4000/100 = 40, while the cost of the edge 2 – 4 will be constant, where L = 45. Hence, all the drivers will incur a cost of 85. However, the cost of the edge 3-4 is less than the cost of the edge 2-4, so drivers will have incentive to start using 1-2-3-4 instead of 1-2-4.
2. On the other hand, if we consider the case where the drivers were equally divided between the paths 1 -> 2 -> 4 and 1 -> 3 -> 4, then the total cost will be the cost of the variable edge x/100 = 2000/100 = 20 plus the cost of the constant edge L = 45. Consequently, the total cost will be 65. Again, this is not Nash equilibrium because the cost of edge 3-4 is less than the cost of the edge 2-4. Therefore, drivers will favor 1-2-3-4 over 1-2-4. Similarly, we can argue that drivers will favor 1-2-3-4 over 1-3-4 because the cost of 1-2 is less than the cost of 1-3.

In order to establish whether Braess’s Paradox is occurring and how to avoid it, we study the result of eliminating the different edges used in the traffic path at Nash Equilibrium; hence we eliminate 1 -> 2, 2 -> 3 and 3 -> 4, one edge at a time. We have the following three scenarios:

```plaintext
Listing 1  Our proposed algorithm

path_NO = 0;
while(path_NO< path.size()-1) {  //store path number
    prev_path = path[path_NO];  //previous vertex number
    next_path = path[path_NO+1];  //next path number
    deledge(prev_path, next_path, node); //delete edge between
    //prev_path and next_path
    for(i=0; i<CAR_NUM; i++) {   //find the Nash equilibriums
        dijkstra(g.vertex_num+1, start, end, node); //propose the dijkstra
        //method
    } if(avg_time2<avg_time1) { //if average travel time gets faster
        route = new_route; //update the routing path
    }
    addedge(prev_path, next_path, node); //add a deleted edge
    path_NO++;
}
```
1. When the city government eliminates edge 1-2 as shown in Fig. 2, the traffic pattern in the case of Nash equilibrium of the new the network is when the cars use the lower route through 3. Consequently the total travel time of each driver becomes 85 minutes, since $4000/100 + 45 = 85$.

2. When the city government eliminates edge 2-3 as shown in Fig. 3, the traffic pattern in the case of Nash equilibrium is when all the drivers balance themselves evenly between the two routes. In this case the total travel time of each driver becomes 65 minutes, since $2000/100 + 45 = 65$.

3. When the city government eliminates edge 3-4 as shown in Fig. 4, the traffic pattern in the case of Nash equilibrium is when all the cars use the upper route through 2. Consequently the total travel time for everyone is 85 minutes, since $4000/100+45 = 85$.

Next, we compare the time cost of the new cases with the time of the original network to find the one with the minimum time. In this case we adopt the traffic pattern of Fig. 3 with edge 2-3 eliminated as it allows the least cost.

Since the occurrence of Braess's paradox is caused by adding new routes to a network and consequently allowing new strategies, we can avoid it by using the proposed method. This is achievable because the edge that causes Braess's paradox is always used in the traffic route.

**SIMULATION**

The simulation program is designed as follows. Each driver seeks to minimize his travel time. Let’s say we have N drivers. The first driver will compare the travel time using the different routes and would choose the route with the least travel time. Then the second driver will also do the same. The simulation program will repeat this consecutively till it finally reaches the Nth driver and finds his best route.

In this simulation, we take the model shown in Fig. 1. It is important to note that the edges with a fixed cost are independent of the number of passing cars; therefore, the cost of using them is constant regardless of the number of cars. Cars using edges with a variable cost ($x/100$ in this case) will suffer a delay proportional to the number of the cars using the same edge.

In this model, the edges 1-3 and 2-4 are insensitive to congestion as they have a fixed cost. However, the edges 1-2 and 3-4 are sensitive to congestion, as they have a variable cost directly proportional to the number of passing cars. When Braess’s paradox occurs, we compare the percentage of reduction of delay with each change in the network.

The simulator is written in C++ and it follows the assumptions below:

![Fig.2 Deleting the edge 1-2 will cause all the traffic to follow the path 1-3-4 in this directed graph.](image)

![Fig.3 In this directed graph, deleting the edge 2-3 will cause the traffic to divide between the paths 1-2-4 and 1-3-4.](image)

![Fig.4 After deleting the edge 3-4 in this directed graph, the traffic will use the route 1-2-4.](image)
The model can be either a directed graph or an undirected graph.

Each driver will follow a strategy to minimize his cost.

Each car has information regarding the route chosen by other cars, in particular, the number of cars using the edges with variable cost.

We carry out two simulations:
1. Simulation of directed graph
2. Simulation of undirected graph

**Simulation of Directed Graph**

In this simulation, we use the directed graph shown in Fig. 1.

Fig. 5 shows the simulation results before applying our proposed method. The graphs show the cost after adding the new edge divided by the cost before adding it. The vertical axis indicates the ratio of delay while the horizontal axis indicates the number of cars. In particular, when the value indicated on the vertical axis surpasses 1, this means that Braess’s paradox is occurring.

Fig. 6 shows the simulation results after applying our proposed method. The graph shows the travel time after applying our proposed divided by the time required when Braess’s paradox is not occurring. As a result, the value indicated by the vertical does not surpass 1. In other words, Braess’s paradox has been avoided.

Fig. 7 shows the reduction in the average travel time after applying the proposed method. The vertical axis is the value obtained by dividing the travel time when Braess’s paradox is avoided by the travel time when the paradox is occurring. The horizontal axis indicates the number of cars. It is clear that average travel time is reduced by a maximum of 25%.

![Fig. 5](image_url)

*Fig. 5* Ratio of delay before applying our proposed method in the directed graph

![Fig. 6](image_url)

*Fig. 6* Ratio of delay after applying our proposed method in the directed graph
SIMULATION OF UNDIRECTED GRAPH

In this simulation, we use the undirected graph model, where the only difference from Fig. 1 is that the edges are undirected. The major practical difference in our case between directed and undirected graphs is that the driver can move from 3 to 2.

Fig. 8 shows the simulation results before applying our proposed method. The vertical axis indicates the ratio of delay caused by adding the edge, while the horizontal axis indicates the number of cars. In particular, when the value indicated on the vertical axis surpasses 1, this means that Braess’s paradox is occurring.

Fig. 9 shows the simulation results after applying our proposed method. As a result, the value indicated by the vertical axis does not surpass 1. In other words, Braess’s paradox has been avoided.

Fig. 10 shows the reduction in the average travel time after applying the proposed method. The vertical axis value is obtained by dividing the travel time when Braess’s paradox is avoided by the travel time when the paradox is occurring. The horizontal axis indicates the number of cars. It is clear that the average travel time is reduced by a maximum of 25%.

THE RANGE WHERE THE PARADOX OCCURS

In this section, we discuss the range of the number of cars where Braess’s Paradox occurs on our undirected network. Let the cost of the constant edge be L, and number of cars be N. In Fig. 1, Nash equilibrium is achieved when the drivers balance themselves evenly between the two routes, then the average travel time is L + (N/2)(1/100) or simply L + N/200.

After adding the edge 2-3, the average travel time (for
Braess’s paradox starts to occur if the average travel time before adding edge 2-3 is better than the new one from the point of view of the driver. Hence more drivers have an incentive to use the new edge, in other words this driver can decrease his total time by changing his route. Consider the case of a single driver who starts to use the edge 2-3, Braess’s paradox occurs if \((N/2)(1/100) + 0 + (N/2 + 1) (1/100) < L + N/200\).

Moreover, Braess’s paradox will continue as long as this inequality is satisfied, \(L + N/200 < 2N/100\).

By solving the two inequalities, we can say that the paradox occurs on condition that the number of cars is \(200L/3 < N < 200L - 2\).

The range where Braess’s paradox occurs in this case is directly proportional to the cost of the constant edge. For various values of \(L\), the range where the paradox occurs in both graph is shown in Fig. 11. The vertical axis indicates the label of edge with constant cost, while the horizontal axis indicates the number of cars.

**DIRECTED VERSUS UNDIRECTED GRAPHS**

Before adding the edge 2-3, during Nash equilibrium, the cars will be divided equally between the routes 1-2-4 and 1-3-4 and hence the cost will be \(L + (n/2)(1/100)\). After adding the edge 2-3 with negligible cost, we will have different scenarios based on the type of the graph.

**DIRECTED GRAPHS**

**Case 1: \(N/100 < L < 3N/200\)**

Because \(L > N/100\), the edges 1-2 and 3-4 will be favorable to 1-3 or 2-4 because of their constantly lower cost. Consequently, in Nash equilibrium, we find all the drivers

---

Fig. 9 Ratio of delay after applying our proposed method in the undirected graph

Fig. 10 Rate of decrease in the incurred delay in the undirected graph
following the route 1-2-3-4 and none of them will have incentive to change. This will trigger Braess’s Paradox where the incurred cost for each driver will be 2N/100.

**CASE 2: N/200 < L < N/100**

The first $x$ drivers will have the incentive to use the same route as in case 1, i.e. 1-2-3-4. This will be the case as long as $x/100 \leq L$. As the number of drivers increases such that $x/100 > L$, the drivers will have an incentive to start using the edges with cost $L$ as well, i.e. edges 1-3 and 2-4. Eventually, at Nash equilibrium, the total cost will be 2L.

**UNDIRECTED GRAPHS**

**CASE 1: N/100 < L < 3N/200**

This is similar to Case 1 for directed graphs discussed earlier. The incurred cost for each driver will also be 2N/100.

**CASE 2: N/200 < L < N/100**

The main difference from Case 2 for directed graphs discussed earlier is that the drivers can use the route 1-3-2-4 in this case. This is because the drivers can freely go from vertex 3 to vertex 2 as the edge 2-3 is not directed. Consequently, the incurred cost for each driver will be 2L in Nash equilibrium.

**CONCLUSION**

The directed graph shown in Fig. 1 and the undirected version of the same graph suffer from Braess’s Paradox if the following inequality holds, $200L/3 < N < 200L - 2$. The total time cost is the same for the directed and undirected cases when Braess’s Paradox is occurring.

In this paper, we propose a method to avoid Braess’s paradox and prove its efficiency through simulation. This method can be used to find the road that should be closed in order to avoid congestion in the network.

**ACKNOWLEDGEMENTS**

We would like to thank the anonymous referee for his valuable comments and suggestions.

**References**

About the Authors

Toshiki Kojima
received the D. E. degree in department of computer science from Gunma University, Gunma in 2012. Since April 2012, he has been with Graduate School of Engineering Master's Program, Gunma University.

Ahmad Bazzi
received his Diploma in Computer and Telecommunications Engineering from the Lebanese University in 2003. In 2011, he received his Master's degree in Computer Engineering from Gunma University, Japan, where he is now a PhD candidate.

Yoshikuni Onozato (Member)
received the D. E. degree in Electrical and Communication Engineering from Tohoku University, Sendai in 1981. Since April 1992, he has been with Gunma University where he is currently a Professor. His research interests lie in the areas related to computer communications. He is a member of IPSJ.