On The Design of a Route Parsing Engine for Connected Vehicles with Applications to Congestion Management Systems

Sam Sinnott[1], Rodrigo Ordóñez-Hurtado[1], Giovanni Russo[2] and Robert Shorten[1,2]

***Abstract*— In this paper we present a new type of congestion management system to regulate congestion around a single link obstruction. This obstruction will be investigated in two cases: a) a regular obstruction and b) an irregular obstruction. The principle innovations of our system are to utilize a route parsing engine to both predict the likelihood of a vehicle being affected by the obstruction and make decisions ahead of time and to use this information to make a *fair* congestion management system.**

I. Introduction

Vehicles are undergoing a transformation from independent bodies operating on a road network, to connected/informed devices, our view of vehicles on our roads is changing. Through vehicle to vehicle (V2V) and vehicle to infrastructure (V2I) communication, the connected car shows a lot of promise for the design of smarter road networks. It is predicted that by 2018, 17% of vehicles in the global fleet will be connected to the internet [1], representing a significant increase from the 10% that were connected as of 2015. This is confirmed by a survey by McKinsey [2] where 13% of respondents said that they are no longer prepared to buy a new vehicle without internet access. This demonstrates that drivers are open to the plethora of new services enabled by connected cars.

While current vehicles can typically be categorised as "passive autonomous" as drivers are being assisted by systems local to the vehicle, future vehicles are likely to be bombarded with information from a multitude of connected devices. In this context, it is very important that the vehicle be able to filter or parse this information, so only the relevant information to the journey is passed to the driver or onboard decision support system. Thus, the development of future vehicles will require them to become cognitive bodies able to filter through a flood of newly shared information and pass only relevant information to the driver. A key innovation in the work presented in this paper is to take a first step in this direction: namely to design an onboard parsing engine that can be used to enhance recently developed congestion management algorithms. Specifically, this paper aims to develop a tool to further improve the ability of contemporary congestion management systems, to give all vehicles an opportunity to access a preferred route, balances rerouted vehicles across adjacent routes, uses feedback control to regulate the flow of vehicles along an obstructed route, and finally uses a parsing engine to filter route closure information only to relevant cars.

[1]Sam Sinnott, Rodrigo Ordóñez-Hurtado and Robert Shorten are with the University College Dublin, School of Electrical, Electronic and Communications Engineering sam.sinnott@ucd.ie, rodrigo.ordonez-hurtado@ucd.ie

[2]Giovanni Russo and Robert Shorten are with IBM Research, Ireland grusso@ie.ibm.com, robshort@ie.ibm.com

A. Problem Statement

The goal of this paper is to design a congestion management system with the aim of re-routing vehicles around a single link obstruction. The system will have the following characteristics: a) it will be capable of re-routing a population of vehicles given opportunity to access preferred routes; b) re-routed vehicles will be balanced across N alternative routes $a_1, ..., a_N$; c) feedback control will be used to regulate the flow of vehicles along obstructed route o; d) using a route parsing engine, the system will filter out information about route closures and pass it only to relevant vehicles. Our interest in single link obstructions (e.g., lane closures due to an accident, and reduced flow in response to planned events such as school opening/closing times) is motivated by the fact that they are potential critical scenarios for the connected cars concerning channel quality, communication range and sensor performance, among many others [3], [4]. Essentially, the system consists of four main components:

- a *route prediction algorithm*, that takes the past behaviour of the user as input and returns the most likely route of the vehicle (i.e. an estimation of the route of the vehicle, given historical data on past trips).
- the output of the route prediction algorithm, together with environmental information (e.g. traffic congestion), is used as input to the *route parsing engine*. It determines whether an obstruction impacts the predicted route, and filters information as it returns only information that will affect the most likely route.
- the *feedback (or network) controller* is used to regulate the number of vehicles passing through the obstructed link o. Taking the measured number of vehicles along the route as an input, it outputs the probability of a vehicle being allocated the obstructed route.
- the *load balancer* is responsible for distributing the vehicles predicted to travel via the obstructed route, but not allocated the resource. It takes the probability of assignment and environmental information as input and determines an alternative route to be allocated.

The two types of single link obstruction (or events in what follows) investigated in this paper are as follows:

- an *irregular obstruction*, which typically does not happen in the same place and can not be predicted. This can be linked to a lane closure due to, e.g., road works.
- a *regular obstruction*, which can be predicted and resources are regularly adjusted during this period. This

can be likened to the closure of a school or an event at an entertainment venue.

The structure of this paper is as follows: Section II outlines the tools used by the system, Section III outlines the proposed system architecture, Section IV discusses the experimental validation of the system, Section IV-C describes the results found, and finally Section V concludes the paper and states the future work.

II. Preliminaries

This section will briefly describe tools used within our design. Specifically we will make use of Markov chains and the PageRank Algorithm in the design of our route parsing engine. Please refer to [5] and [6] for further details.

A. Markov Chains

A Markov model (or Markov chain in what follows) is a tuple $\langle S, \mathcal{A}, T \rangle$, with S being a finite set of states $(s_1, \ldots, s_n)$, $\mathcal{A}$ being a finite set of actions, and T being the transition function $T : S \times \mathcal{A} \times S \to \mathbb{R}$. The transition function $T(s_i, a, s_j)$ models the probability that the system goes from state s_j to s_i when action a is executed, i.e. $T(s_i, a, s_j) = p(s_i|s_j, a)$. Given a Markov model and an initial state distribution, say π, then the future state distribution arising from the sequence of actions can be predicted. Specifically, let $p^t(s_i)$ be the probability of being in state s_i at time t, then $p^{t+1}(s_i) = \sum_{s_j \in S} p^t(s_j) T(s_i, a^t, s_j)$.

In the rest of the paper we will make use of Markov models to build the route prediction algorithm presented in Section III. See [7], [8], [9] for similar ideas in this field.

B. Ranking Algorithm

A ranking algorithm allows for objects in a system to be ranked against each other. The type of ranking algorithm we will use to develop our system is the so-called PageRank algorithm [6]. Originally made popular by Google™, this algorithm is typically implemented on webpages/nodes and ranks the relationship between other pages/nodes.

Intuitively, the PageRank algorithm $PR(\bullet)$ computes the *importance* of a given node $\bullet$ in a graph G, where the input to the algorithm is essentially the topology of G. Specifically, let A be the adjacency matrix of G, the ranking for the i-th node is then $r_i := PR(d, r_j, A) = \left((1-d) + d \sum_{j \in N_i} r_j\right)$, where N_i is the set of nodes linked to (i.e. pointing to) node i, and d is a design parameter (a damping factor) which in the clasical PageRank algorithm for web pages is typically set to 0.85. In our study, inspired by [10], d will be set to 0.93. Hereafter, we will use a version of the PageRank algorithm (applied to the edges of a graph rather than nodes) to rank possible routes that a vehicle could take.

III. The Proposed System

The proposed system is presented in Fig. 1. We will describe the required modules and algorithms in the following subsections. However, for the sake of brevity, we omit the theoretical results beyond our algorithms.

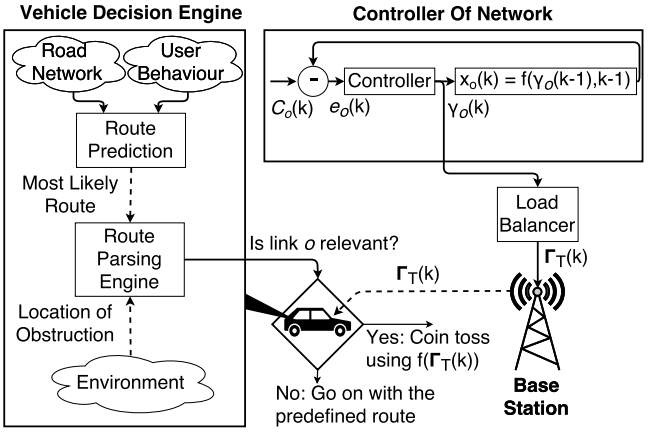

Fig. 1: System Architecture. $C_o(k)$ is the desired capacity of the obstructed link o; $x_o(k)$ is the number of vehicles on the obstructed link o (i.e. the controlled variable of the obstructed route); $e_o(k) := C_o(k) - x_o(k)$; $\mathbf{\Gamma}_T = \gamma_o(k) \cup \mathbf{\Gamma}_{alt}(k)$, where $\gamma_o(k)$ is a variable affecting o, and $\mathbf{\Gamma}_{alt}$ is a set of parameters calculated by the load balancer; and $f(\gamma_o(k), k)$ models the time evolution of the capacity along o.

Summarizing from Fig.1, the route predictor calculates the most likely route for the user to take given the users previous journeys. Based on this, the route parsing engine determines if the predicted route is affected by the obstruction. Then, a time-varying signal, say $\gamma_o(k)$, is calculated by the feedback controller and it is passed to the load balancer. This module, in turn, calculates a set of probabilities, i.e. one probability for each alternative route, and broadcasts this set to the vehicle. The vehicle, in turn, flips a coin to decide which alternative route it will follow.

A. Route Prediction

The goal of the Route Prediction algorithm is to compute the most likely route that the vehicle will take. This output is then used by the Route Parsing Engine to detect obstructions along the route. The route predictor gathers data about the past trips of the user and creates a picture of where the user typically goes along a journey. In this paper, a journey is defined through a graph, G_T, that connects origins and destinations [11]. The nodes of the graph will be road intersections and links will be roads linking different intersections. From the functional viewpoint, the route predictor can be broken down into three sub-modules: 1) Edge Ranking Routine, 2) Edge Weighting Routine, and 3) Prediction Routine.

1) Edge Ranking Routine: The key idea of this routine is to characterize the *importance* of a given route from a topological viewpoint. This is done by ranking the edges (and hence the directions) taken by a driver during a trip. The Edge Ranking Routine described here then takes as input the topology of G_T and ranks its edges. Specifically, this is done by applying the PageRank algorithm to the edges of G_T, rather than to its nodes. Let $r_{i,j}$ denote the ranking of the edge (i, j), i.e. the edge linking nodes i and j, and let

E_T denote the edge-adjacency matrix of G_T. Then we have

$$r_{i,j} := PR\left(d, r_{m,n}, E_T\right) = \left((1-d) + d \sum_{(m,n)\in N_i^e} r_{m,n}\right), \tag{1}$$

where N_i^e is the set of edges adjacent to (i.e. pointing to) edge (i,j).

2) Edge Weighting Routine: The key idea of this routine is to characterize the edges of the graph G_T based on the *habits* of the user, i.e. based on past trips. The edge weighting routine computes weights for each of the edges of G_T using

$$f_{i,j} := \frac{\textit{Number of trips from i to j}}{\textit{Total Number of trips departing from i}}. \tag{2}$$

Edges that have been taken more often have higher weights.

3) Route Prediction Routine: Edge rankings and weights are provided as inputs to this routine, which performs the route prediction. In order to illustrate this routine, let $tr\left(i,n\right)$ be a *tree* of width n generated from the i-th node of the graph G_T. We will denote by $br\left(i\right)$ the i-th branch of such a tree. The route prediction is then achieved by computing the quantity

$$R_j := \sum_{(m,n)\in br(j)} f_{m,n} r_{m,n}. \tag{3}$$

Note that each R_j is associated to a given branch $br\left(j\right)$ (i.e. to a window of n subsequent edges along the possible routes). Then, the predicted route is the one having the maximum R_j. The window size chosen for this problem is $n = 5$. From simulations, this value offered a good compromise between computational efficiency and prediction accuracy. The macro steps for route prediction are shown in Algorithm 1.

B. Route Parsing Engine

The objective of the route parsing engine is to parse information coming from the network environment as a function of the route predicted by the prediction algorithm. Specifically, this module determines if the vehicle in question will be affected by an obstruction. If the predicted route is seen to be via the obstructed route, then this information is provided to the feedback controller and load balancing module. This can be seen in Algorithm 2.

Algorithm 1 Route Prediction Algorithm

procedure ROUTE PREDICTION
  Find current edge i based on car location
  Compute the tree $tr\left(i,n\right)$
  for each edge (i,j) in $tr\left(i,n\right)$ **do**
    Compute edge ranking $r_{i,j}$ by means of (1) and weights $f_{i,j}$ by means of (2)
  for each branch $br\left(i\right)$ in $tr\left(i,n\right)$ **do**
    Compute branch ranking R_i by means of (3)
  return route with the largest R_i

Algorithm 2 Route Parsing Engine Algorithm

procedure PARSING ENVIRONMENTAL INFORMATION
  Get the route with largest R_i from Prediction module
  Get environmental information about obstructions
  if obstruction along predicted route **then**
    return relevant information

C. Feedback Controller

To regulate the number of cars along the obstructed link o, a closed loop feedback control system has been implemented. This method of regulation was chosen due to the nature of an obstruction. Specifically, in the case where the flow rate of the obstructed route was being controlled and becomes zero, queues would form and the congestion management system would not achieve its objective.

What differentiates this work from others such as [12] is that, firstly, in this instance the number of vehicles on the route are being controlled in a token/buffer like fashion rather than the control of the flow of vehicles through a regulated section of road. Secondly, the investigation in this paper is using a parsing engine to specifically determine which vehicle should be re-routed or be allocated the resource.

1) Allocation Through An Irregular Obstruction: Here we assume the obstruction does not happen regularly in the same place and, as a result, the allocation of resources will be done in a way to maximise the quality of service for all users. The controlled variable $x_o\left(k\right)$ in this case is the measured rate of vehicles on link o, the $C_o\left(k\right)$ is the correspondent desired value, and the type of controller used is an On-Off controller with a dynamic described by

$$\gamma_o\left(k\right) = \begin{cases} 0, & \text{if } e_o\left(k\right) < 0, \\ 1, & \text{otherwise,} \end{cases} \tag{4}$$

where $P_o\left(k\right) = \gamma_o\left(k\right)$ is the probability of a vehicle being allocated though the regulated link o, and only depends on $e_o\left(k\right) = C_o\left(k\right) - x_o\left(k\right)$. Note that in this investigation, the probability of assignment is presented as "certainties" (i.e. 0 or 1) as a result of the use of an On-Off controller. In applications which desire the probability of assignment to be adjusted with respect to the amount of resources available, other types of controller may be used (e.g., a PID controller, which will allow for less bursty traffic to be passed through the obstructed route); this, however, is beyond the scope of this investigation.

2) Allocation Through A Regular Obstruction: In the case of a regular obstruction, the period of time which we would like to balance the average user allocation is larger and, as a result, merits a more tailored design. Taking inspiration from [13], where access to parking facilities was managed in a way to optimise the benefit to a population, we now apply similar ideas to govern the access to the obstructed link. Note that, optimization in [13] is achieved when the utility functions achieve consensus. Thus, by selecting the utility functions that govern the access of each user, in an

appropriate manner, then a range of optimality criteria can be realised.

This is achieved by first using the same feedback controller structure shown in Fig. 1, but here $\gamma_o(k)$ is not a probability of allocation being distributed to each vehicle, but a gain which will scale the users probability of allocation to this obstructed route. Now the controller updates $\gamma_o(k)$ as a function of $e_o(k)$ by

$$\gamma_o(k) = \gamma_o(k-1) - \alpha * e_o(k), \tag{5}$$

where $e_o(k) = C_o(k) - x_o(k)$ denotes the capacity available along the route, α is a scaling constant, and γ_o is bound by $0 \leq \gamma_o(k) \leq 1$. For each user, the historic probability of being allocated the resource will be determined by the number of times a user has been granted and refused their preference. This can be described by

$$H(k) = \frac{\bar{Y}_i(k)}{{f_c}'_i\left(\bar{Y}_i(k)\right)}, \tag{6}$$

where $\bar{Y}_i$ denotes the average allocation of resources for user i, and ${f_c}'_i\left(\bar{Y}_i(k)\right)$ is a concave cost function for user i which will be minimised to reduce the number of times the user is re-routed. In this implementation, the cost function was chosen to be $f_c(z) = z^4/4$ and the scaling factor $\alpha = 0.01$.

Here, the probability of allocation along the obstructed link o, $P_o(k)$, is found by

$$P_o(k) = \gamma_o(k) * H(k). \tag{7}$$

Remark: Eq. (7) allows for the privacy of a user to be preserved as each vehicle utility function $H(k)$ can be held locally and $\gamma_o(k)$ can be broadcasted to network for decisions on probability of assignment. When P_o is found, this value is then pitched against a coin toss to stochastically decide if the resource will be allocated to the user. If after the coin toss, the obstructed route is not chosen, the information from the load balancer will be used to re-route the vehicle. The stochastic nature of the coin toss avoids predictability and allows for fairness to be created as no one user will have a higher priority to the resources than others.

D. Load Balancer

The load balancer is responsible for calculating the probabilities of alternative routes to the obstructed route, which will be sent to the car but only going to be used in the event that the obstructed route is relevant and cannot be allocated. For this, and an observation of the environment is carried out.

This observation is namely to identify if there is capacity on the obstructed route o to accept this vehicle, and, if required, to calculate $\mathbf{\Gamma}_{alt}(k) = \{\gamma_{a_i}(k)\}_{i=1,\ldots,N}$, i.e. the set of probabilities to take N alternative routes between their origin and likely destination. In the case that $x_i(k)$ (i.e. the number of vehicles along route i) is different than zero for all i, then the load balancing of re-routed cars is given by

$$\gamma_{a_i} = \frac{1/h_i}{\sum_{j=0}^{N}(1/h_j)}, \text{with } h_i = \frac{x_i(k)}{\sum_{j=0}^{N}(x_j(k))}, \tag{8}$$

where h_i can be seen as the proportion of the total load which is passing through route i, otherwise (i.e. when $\exists x_i(k) = 0 \forall i$) the probability of assignment is determined by

$$\gamma_{a_i} = \begin{cases} \frac{1}{\left|\{x_i(k)\,|x_i(k)=0\}_{i=1,\ldots,N}\right|}, & \forall i \text{ s.t } x_i(k) = 0, \\ 0, & \text{otherwise,} \end{cases} \tag{9}$$

where $|\bullet|$ is the cardinality of the set $\bullet$, this is that when the flow through any exit i is zero, the probability of assignment for that exit is equal to the the inverse of the total number of resources where $x_i(k) = 0$. Finally, $\mathbf{\Gamma}_T(k) = \{\gamma_o, \gamma_{a_1}, ..., \gamma_{a_N}\}$ is sent to the vehicle approaching the link. Based on γ_o, the vehicle calculates the probability of accessing the obstructed link. The vehicle then flips a coin to determine which link it will take.

IV. Experimental Validation

A. Simulation Software

The software tool used to simulate and validate this design is SUMO [14]. Developed at the Institute of Transportation Systems at the German Aerospace Centre (DLR), SUMO is an open source microscopic simulation platform that allows for the simulation of vehicles on large networks. Modelling not only vehicular dynamics and vehicle following models, SUMO also allows for a realistic reconstruction of a problem and implementation of a vehicle to infrastructure solution.

SUMO simulations are comprised of three layers:

1) Network Layer - The simulated road infrastructure,
2) Vehicle Layer - The cars interacting in the simulation,
3) Route Layer - The routes traced by the simulated cars.

The chosen network (depicted in Fig. 2), which this congestion management system has been implemented on, is the area of Crumlin in Dublin 12, Ireland. This network was imported from www.OpenStreetMap.org, an open source, community-driven mapping service. In Fig. 2, we can see the point of obstruction highlighted by a caution sign, and the detection range depicted by a blue circle. The detection range around the incident is where vehicles will be assessed for their likelihood of passing via the obstruction. In this investigation, the radius of this area was been set about 575m, chosen like this to ensure that the delay between vehicles being allocated resources and vehicles clearing the obstruction was short, and to allow for more alternative routes to be taken.

Fig. 2: Simulation Network: Crumlin, Dublin 12, Ireland.

The journeys for this simulation were chosen to be between 17 origins and 17 destinations on either side of the network. The routes used were generated by using SUMO's *duarouter* function, which finds the shortest path between the specified origin and destination.

B. Implemented Algorithm

To implement the route parsing algorithm, TraCI (a Python-based interface to access running simulations in SUMO) was used to communicate between the infrastructure and simulated cars. The first step of the route parsing algorithm outlined in Section III was to generate the Markov transition matrix for each origin by observing the edges that a vehicle leaving this origin took for each of the journeys to each of the destinations. These observations of edges vehicles passed on were counted, and the most frequently used edges recorded. The row state vector was then defined by randomly generating the number of times a vehicle travelled between each origin and destination.

The obstruction was created by selecting an edge used by a large number of journeys and determining an appropriate detection radius for which oncoming vehicles would be assessed. In this simulation, the reference $C_o\,(k)$ was set to three vehicles at any one time (i.e. the maximum number of simultaneous cars allowed within the link is equal to 3) and the obstruction itself was modelled by a significant reduction in speed along the edge, where the change in speed limit was from 50 km/h to 18 km/h. This is a typical speed change for roads in the surroundings of a traffic incident.

At the point where an incoming vehicle reaches the affected zone, the route prediction (Algorithm 1) and route parsing engine (Algorithm 2) start to determine if the vehicle is likely to be affected by the incident. At this point the probability of the vehicle becoming in contact with the incident will be known, and the feedback controller of Section III-C will attempt to regulate the capacity of the obstructed segment. Vehicles are then re-routed through the load balancer algorithm of Section III-D. Specifically, the algorithm will assess the current load on the affected link and adjacent links, and will determine the route to be taken by the car.

Re-routing vehicles which have not been allocated the affected link will depend on the origin and the destination of the travelling vehicle. The objective of the load balancing algorithm is to balance the number of vehicles on each of the adjacent alternative routes and to re-routed vehicles in a way that reduces further congestion. This is achieved by assessing the load on links of the alternative routes around the incident and allocating alternative resources according to current loads.

C. Numerical Results

Here we discuss the results obtained for both the irregular and regular obstructions.

1) Irregular Obstruction: The irregular obstruction was simulated for a period of 4 hours, where vehicles within the affected radius were assessed to check if they would be travelling via the obstruction. In the cases where the vehicle was likely to come in contact with the obstruction, the vehicle was routed according to the environments current loads.

The results of the irregular single link obstruction can be seen in Fig. 3. The top panel of the same figure shows the flow of vehicles through the incident route and three of the alternative routes without load balancing or route prediction. It can be seen that the incident route carries a large amount of traffic along it as it is the main link for a large number of vehicles in the simulation.

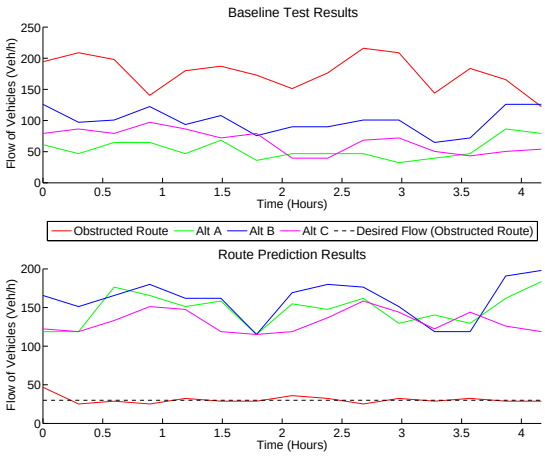

Fig. 3: Results for the irregular obstruction.

Fig. 3-bottom, instead shows the results of the system using the route prediction and load balancing algorithm. It can been seen that the traffic flow through the obstructed route has been significantly reduced according to the specifications and is seen to be regulated at an average of 30 vehicles per hour. This shows that the number of vehicles which has passed the obstruction has successfully been regulated over time. It can also be seen that the three alternative routes have become closely matched as a result of the load balancing. Alternative route B, which was previously at an average of 54 vehicles per hour, rose to an average of 147 vehicles per hour. This shows that a large amount of the traffic required to be distributed along the alternative routes has been achieved, and routes with lower congestion have taken more of the load from the obstructed route.

2) Regular Obstruction: The investigation of the regular obstruction was carried out over the same daily period of 4 hours as the control in the irregular obstruction, and was modelled with a similar regular load. It was assumed that the area around the obstruction is a residential area which contains a school and the vehicles competing for access would typically be using the area every day. The focus here is to repeat the simulation over the course of a year and incorporate fairness into the allocation of resources, by allowing each user have equal opportunity (on average) to access the resources over the course of the investigated time. Due to the large timeframe used, a macroscopic approach was taken and the simulation was carried out in MATLAB®. Each daily investigation was broken up into 15 minute incre-

ments and a running average of the flow of vehicles through the obstruction per hour was calculated. This approach was chosen to account for bursty arrival of traffic within the observed interval. Using a running average allows for this type of traffic to be filtered out over time, which in turn allows for an assumption that the equilibrium probabilities over the interval of time of interest do not vary too much from a given interval to the next.

When a user is competing to use the affected link, their history is queried, and the average number of times the user has been allocated the resource along with the flow of vehicles on the route during this period is then used to determine the probability of allocation. Once this probability is known, a coin toss is carried out as described in [12], which competes with the historic likelihood of being allocated the resource to add a stochastic element to the allocation of the resource.

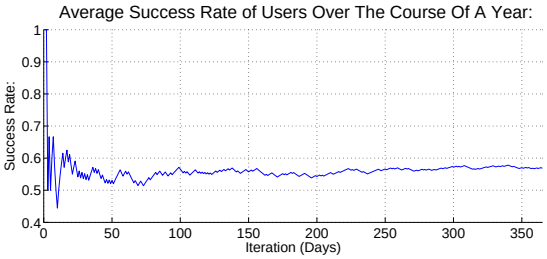

Fig. 4: Regular Obstruction: Utility fairness results.

It can be seen from Fig. 4 that the equilibrium probability for all users using cost function $f_c(z) = z^4/4$, found in Section III-C.2, is 57% over the course of one year. This shows that over time, all users using this cost function have been allocated this resource approximately an equal number of times, allowing for a suggestion of fairness in the solution. While this is not proved in this paper, it has been discussed in [15] and is suggested in the results presented.

V. Conclusions and Future Work

This paper has successfully investigated the design of a congestion management system with the following characteristics: a) the system is capable of rerouting a population of vehicles giving *fair* access to the preferred route; b) re-routed vehicles are balanced across adjacent alternative routes; c) the system uses feedback control theory to regulate the flow of vehicles along the obstructed link; and d) using a route parsing engine, the system filters out information about route closures and pass it only to relevant vehicles. The results presented in Section IV-C confirm that our approach has been an effective solution to the problem of a single link obstruction.

The investigation was carried out on two different types of obstruction, a regular and irregular obstruction. The former case showed that vehicular flow along the obstructed link could be regulated, and the rest of the load balanced across alternative routes. In the second investigation, the case of the regular obstruction showed that when an obstruction can be predicted, vehicles can be allocated access to this resource in a *fair* way over a longer period of time. It should be noted here that the inclusion of a route parsing engine allowed for the re-routing of vehicles to be done in a time effective manner, which allowed for more alternative routes to be utilised.

Future work is to develop the system further to update users transition matrices in real time and to study, via e.g. nonlinear contraction theory [16], stability properties of the resulting closed loop dynamics. Additional future work concerns investigating about how this system could be modelled using real vehicles. Work has previously been carried out in papers such as [12] to develop a hardware in loop platform for SUMO, allowing for a real vehicle to become embedded in a SUMO simulation. This platform would allow for the validation of the route prediction algorithm by creating a model of a users driving behaviour, and assessing if the algorithm can predict where a user would typically go. This test of user behaviour would allow for great insight into the 93% predictability proposed by [10]. A video demonstrating the work in [12] can be found at [17].

References

[1] Volkswagen ViaVision, "Shaping the future of mobility," 2015.

[2] McKinsey, "Whats driving the connected car," tech. rep., http://goo.gl/8I4Lzx, 2014.

[3] R. Meireles, M. Boban, P. Steenkiste, O. Tonguz, and J. Barros, "Experimental study on the impact of vehicular obstructions in VANETs," in *2010 IEEE VNC Conference*, IEEE, 2010.

[4] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[5] D. G. Luenberger, *Introduction to Dynamic Systems*. Wiley, 1979.

[6] C. Moler, *Experiments with MATLAB*, ch. 8 - Google PageRank. Mathworks, 2011.

[7] M. Faizrahnemoon, A. Schlote, L. Maggi, E. Crisostomi, and R. Shorten, "A big-data model for multi-modal public transportation with application to macroscopic control and optimisation," *International Journal of Control*, vol. 88, no. 11, pp. 2354–2368, 2015.

[8] D. F. Gleich, "PageRank beyond the web," *SIAM Review*, vol. 57, 2015.

[9] A. Schlote, E. Crisostomi, S. Kirkland, and R. Shorten, "Traffic modelling framework for electric vehicles," *International Journal of Control*, vol. 85, no. 7, pp. 880–897, 2012.

[10] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.

[11] C. Godsil and G. F. Royle, *Algebraic graph theory*, vol. 207. Springer Science & Business Media, 2013.

[12] W. Griggs, R. Ordóñez-Hurtado, E. Crisostomi, F. Häusler, K. Massow, and R. Shorten, "A large-scale SUMO-based emulation platform," *IEEE Transactions on ITS*, vol. 16, pp. 3050–3059, Dec 2015.

[13] A. Schlote, C. King, E. Crisostomi, and R. Shorten, "Delay-tolerant stochastic algorithms for parking space assignment," *IEEE Transactions on ITS*, vol. 15, pp. 1922–1935, Oct 2014.

[14] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO-simulation of urban mobility," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3&4, 2012.

[15] W. Griggs, J. Y. Yu, F. Wirth, F. Husler, and R. Shorten, "On the Design of Campus Parking Systems With QoS Guarantees," *IEEE Transactions on ITS*, vol. 17, pp. 1428–1437, May 2016.

[16] G. Russo, M. di Bernardo, and E. D. Sontag, "Stability of networked systems: A multi-scale approach using contraction," in *49th IEEE CDC Conference*, pp. 6559–6564, Dec 2010.

[17] S. Sinnott, *UCD Smart Transport H.I.L.* http://goo.gl/meZU7o.