A Resource Allocation Technique for VANETs Inspired to the Banker's Algorithm

Walter Balzano, Erasmo Prosciutto, Biagio Scotto di Covella, and Silvia Stranieri

Abstract With the fast growth of the number of vehicles on our roads, the traffic congestion problem is becoming an issue in big cities. This work is inspired by a known algorithm, the banker's algorithm, used in operating systems to handle the resource allocation to processes. By following this lead, we treat vehicles like processes making requests and roads as resources to be allocated, and we provide an algorithm to manage the vehicle distribution over the available paths so to reduce the traffic congestion.

Key words: VANETs; traffic congestion; resource allocation; banker's algorithm.

1 Introduction

The population in cities has been continuously increasing since 1985, as the study of [16] highlight, and it is supposed to be still increasing for several years. As a consequence, the number of vehicles on our roads increases as well.

According to what shown in [15], several European cities have a mileage increased over the 50% due to the traffic congestion. In such a scenario, vehicular ad hoc networks, VANETs for short [5], play an important role. The high potential of such a framework pushes researchers into employing them to improve the viability quality.

Let us recall that VANETs are networks made of vehicles able to communicate to each other via broadcasting, by performing a V2V information exchange, if they are close enough, or a V2I one, by using some road infrastructure as a bridge. The concept of "close enough" is defined by the received signal strength indicator, which

University of Naples, Federico II
Balzano W. e-mail: wbalzano@unina.it · Prosciutto E. e-mail: e.prosciutto@studenti.unina.it · Scotto di Covella B. e-mail: b.scottodicovella@studenti.unina.it · Stranieri S. e-mail: silvia.stranieri@unina.it

measures if the signal power between vehicles is strong enough to perform a direct communication.

The traffic congestion is one of the heaviest problems affecting our cities but it is also one of the main challenges in VANETs research field.

In this work, we provide an algorithm to reduce the traffic congestion in VANETs, by defining a revisited version of a known algorithm for resource allocation in operating systems, the banker's algorithm. In the same way the standard algorithm optimise the distribution of the available resources in the operating systems among the processes that need them, the revisited algorithm distributes vehicles among the available paths between a starting and a destination point, so to optimise the roads occupation and improve the viability quality.

Outline: The rest of the paper is organised as follows: Section 2 provides an overview of the literature about algorithmic approaches to the traffic congestion problem, as well as the main phases of the banker's algorithm that will be used in the sequel; in Section 3, we map the resource allocation problems of operating systems to the vehicle distribution over paths in VANETs, by providing the algorithm in Section 4; evaluations and simulations of the proposed solution are shown in Section 5; finally, in Section 6, the conclusions of the work are provided.

2 Background

In this section, we analyse the literature on VANET's traffic congestion management and banker's algorithm, then we recall the standard version of the algorithm.

2.1 Related Works

The traffic congestion management is a topic largely studied from researchers in VANETs research field, see [14] for a survey. Authors of [4], for instance, experimented a behavioural clustering to handle the congestion on the roads. Also in [7], they relies on clustering techniques to manage some critical aspects in VANETs, while in [1, 3, 2], they focus on transports.

In particular, the banker's algorithm has been widely studied for improvements, like they did in [8, 9], or even for smart home applications like in [17], deadlock avoidance [6], resource allocation [10], but it has never been employed for VANET management purposes, which we actually do in this work.

Precisely, in this work we use such a known algorithm to improve the traffic congestion and distribute vehicles among available paths.

2.2 The Banker's Algorithm

One of the main challenging aspects in operating systems is the process management and, in particular, the deadlock prevention. This is a situation that may occur during the resource assignment process. When two different processes ask for the same request at the same time, they can block each other waiting for the resource to be released. This is the reason why several algorithms for deadlock prevention have been provided to handle the resources distribution among the active processes. One of them is the banker's algorithm [6], developed by Edsger Dijkstra in 1965.

The algorithm has been inspired by the loan model used between the banks and the customers, that guarantees not to expire the available economic resources. In operating systems, the processes act like customers, and the resources as money.

Let us recall that a secure state is a state in which all the requests can be satisfied through a resource assignment leading to a correct termination of the involved processes. In other words, a state is said to be safe if the system is able to allocate resources to each process (up to its maximum) in a certain order and prevent a deadlock occurrence.

Figure 1 shows the main phases of the algorithm in a diagram [1].

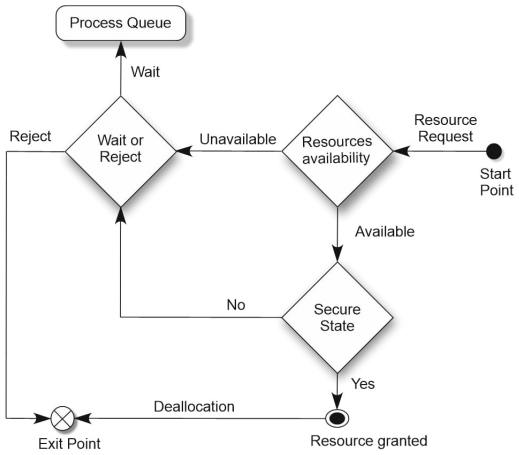

Fig. 1 Flow chart of the banker's algorithm.

Following the figure 1, we can identify three types of paths that give different response to the request for resources:

1. **Granted Resource Path**: $ResourceRequest \rightarrow Available \rightarrow Yes \rightarrow Deallocation$
2. **Non-Granted Resource Path**: $ResourceRequest \rightarrow Unavailable \rightarrow Reject$
3. **Waiting-Queue Path**: $ResourceRequest \rightarrow Unavailable \rightarrow Wait$

[1] In the figure 1, subsequent to the Process Queue structure, the remaining part of the flow was intentionally not provided, thus considering the structure as a 'Black-Box'.

Additionally, an explanation of all the elements in the diagram is here provided.

- *Resources Availability*: is a logical control flow of available resources; it deals with determining whether or not the requested amount of resource is available.
- *Secure State*: is a safe state control logic flow; it is responsible for determining, with appropriate simulation, whether or not the system remains in a safe state with a resource concession.
- *Wait or Reject*: is a logic flow controlling the pause or rejection status; it deals with whether a resource request can be placed in a priority queue or whether it should be rejected.
- *Process Queue*: is a structure containing all processes categorised as ”paused”, i.e. waiting for resources to be released by other processes.
- *Resource Granted*: is a state in which the request has passed both of the previous control flows and has been approved for obtaining the resource.

3 Isomorphism with the congestion problem in VANETs

In this work, we want to handle the traffic congestion problem in the banker's-algorithm-style.

Similarly to the resource allocation in operating system, in VANETs vehicles act like processes, and the resources are the roads. Moreover, in the same way the processes ask for, and then release, the resources, vehicles leave the occupied roads after a certain time, by letting them available for other vehicles.

From this point of view, allocating a large part of vehicles to the same route, initially considered to be the best, would cause congestion, making it available only to a small portion of users. Congestion comes from non optimal allocations of vehicles to the available routes, assigning them a number of vehicles too high to be handled through their capacity.

According to the VANET configuration, some routes can be considered optimal, meaning that it has the right combination of traffic load and mileage time needed to reach the required destination. As vehicles are gradually assigned to the optimal route, such a route starts being almost overcrowded, hence the algorithm decides to redirect the next users to other routes that can be considered optimal in the new configuration.

According to the described isomorphism among the two problems, the banker's algorithm can be used to optimise the traffic flow according to the requests made by the vehicles.

4 VANETs Banker's Algorithm

Let us imagine a user request to reach a point B starting from a point A. The revisited version of the banker's algorithm has a twofold role:

- It checks the paths linking A and B and their saturation levels;
- It forecasts the traffic congestion evolution, by modifying the path to follow to reach B at run time.

The revisited banker's algorithm has two main phases:

1. Pre-conditions acquisition;
2. Secure state verification.

In the following, we are going to analyse deeply the behavior of the algorithm in these phases.

4.1 Pre-conditions acquisition

In order to avoid traffic congestion, the algorithm takes into consideration only a percentage of the maximum capability for each road, leaving the remaining part for users that do not use navigation applications. Such a percentage constitutes a parameter that has to be provided as input to the algorithm.

The information related to how many vehicles can be accepted for each road is static, at the beginning, and it is obtained from statistics, but then it should be clearly updated dynamically, as well as the network evolves.

The real-time available capability of each road is updated through the information received by the GPS devices, supposed to be provided to each vehicle.

In any case, in the event of any unavailability, perhaps linked to the GPS's missing or malfunctioning, a strategy based on maximum and average travel times will be adopted in order to guarantee the proper release of the resource.

Let us now formalise the variables we will need to take into consideration:

1. **Max. Road Capacity** (M.R.C.): indicates the maximum capacity of a road to hold vehicles (*we assume a road to have a max percentage of 100% capacity*).
2. **Max. Manageable Road Capacity** (M.M.R.C.): indicates the maximum capacity of a road to hold vehicles, considering some users (*that we can not map*) who may not use any navigation application.
3. **Max. Assignable Resource** (M.A.R.): indicates the limit of maximum amount of resource that can be assigned, until the algorithm decides to redirect it to another path.
4. **Road Availability** (R.A.): indicates the percentage of real-time road availability.

We want to remind that, similarly to what we have explained in Section 3, in this scenario caring for the best possible management of the release of a route is fundamental for the correct assignment of values to the R.A. and M.A.R. variables, and at the same time, for the accurate routing of vehicles to the optimal route.

We will now provide an example that we hope will perfectly illustrate the possible problems that a bad management of that think would arise.

Let us assume that we have two routes, X and Y, both with a certain M.R.C., M.A.R. and R.A., and a vehicle that requires access to route X. In addition, the driver is equipped with a GPS device that is not fully functional and the system does not use any time-based resource release policy. Access to route X is approved and the vehicle begins its transit within the mentioned route. On average, according to statistics, it should take the vehicle 25 minutes to travel that route. After 30 minutes, the vehicle leaves the route but the GPS continues to report its position within it. Despite the vehicle leaving, the resource is not released and will remain unavailable until the next possible update of the GPS device. Repeating this pattern for several users would lead to a point where the algorithm would no longer have two choices (X and Y), but only Y, as the resources of the first route are all assigned and never released.

Now instead, we will provide a VBA simulation.

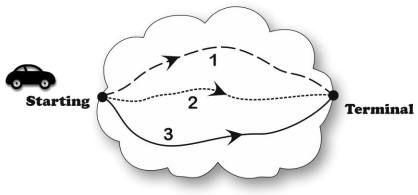

Fig. 2 VBA simulation with 3 possible routes

Table 1: (a) Roads parameter's percentage and (b) roads availability percentage with estimated travel time

(a)

	M.R.C.	M.M.R.C.	M.A.R.	R.A.
Route 1	*100%*	*30%*	*25%*	*56%*
Route 2	*100%*	*45%*	*40%*	*78%*
Route 3	*100%*	*20%*	*15%*	*45%*

(b)

	R.A., trip-time 1	R.A., trip-time 2	R.A., trip-time 3	R.A., trip-time 4
Route 1	*97-100%* *7 minutes*	*81-96%* *12 minutes*	*61-80%* *32 minutes*	*0-60%* *50 minutes*
Route 2	*96-100%* *3 minutes*	*86-95%* *12 minutes*	*71-85%* *18 minutes*	*0-70%* *34 minutes*
Route 3	*81-100%* *5 minutes*	*65-80%* *8 minutes*	*51-65%* *18 minutes*	*0-50%* *25 minutes*

The Figure 2 shows a simulation of the algorithm. In particular, following the arrival of the user at the "starting" point, the algorithm verifies a series of conditions in order to be able to choose the best route to assign him in order to reach the "terminal" destination.

In the first table (1(a)) we have an example of data of type M.R.C., M.M.R.C., M.A.R. and R.A. for the three possible routes, while in the second table (1(b)) we have examples of data based on R.A., where for each percentage range an average travel time is indicated.

The VBA algorithm of Section 4 processes the data shown in Table 1(a), which will allow it to choose the optimal route, corresponding to the one with the highest availability.

4.2 Secure state verification

In the "Secure state verification" step, the algorithm makes a simulation of the secure state (Section 2). Precisely, the algorithm checks whether, by executing a vehi-

Algorithm 1 Secure State Verification

Input: The data structure containing *Roads*
Output: The *optimalRoad* to remain in the secure state

1: $maxAvailability \leftarrow 0$
2: $flag \leftarrow false$
3: $optimalRoad \leftarrow null$

4: **for** $index = 0 \rightarrow$ number of $Roads - 1$ **do**
5: $roadAvailability \leftarrow searchRealTimeAvailability(Roads[index])$
6: **if** $roadAvailability > maxAvailability$ **then**
7: $maxAvailability \leftarrow roadAvailability$
8: $optimalRoad \leftarrow Roads[index]$
9: **end if**
10: **end for**

11: $mmrc \leftarrow searchMMRC(optimalRoad)$
12: $mar \leftarrow searchMAR(optimalRoad)$

13: **while** $flag \neq true$ **do**
14: **if** $maxAvailability > 0 \wedge mar > 0$ **then**
15: **if** $maxAvailability - UserWeight < 0 \vee mar - UserWeight < 0$ **then**
16: $optimalRoad \leftarrow searchAlternativeRoads(optimalRoad, Roads[])$
17: **else if** $maxAvailability - UserWeight > 0 \wedge mar - UserWeight > 0$ **then**
18: $modifyRA(UserWeight, optimalRoad)$
19: $flag \leftarrow true$
20: **end if**
21: **else**
22: $optimalRoad \leftarrow searchAlternativeRoads(optimalRoad, Roads[])$
23: **end if**
24: **end while**

25: **return** *optimalRoad*

cle/road association, that road exceeds its maximum allowed capacity, bringing the system into an unsafe state. In case this happens, the algorithm will make use of an additional function that searches for and returns an alternative road to get to our destination. As can be seen, with Algorithm 1, the heart of the algorithm is enclosed in a while loop: the latter allows us to find a way to reach our destination. In particular,

use is made of a flag variable that, if it has value *false*, implies that the optimal road has not yet been found, otherwise it is set to *true* to exit the while loop and return the road that the user will have to take.

The algorithm also relies on five auxiliary functions to perform its functionality to the best of its ability. In particular:

- the function $searchRealTimeAvailabilty()$ returns the current availability of the road in question;
- the function $searchAlternativeRoads()$ aims to return alternative roads to the currently optimal one, when the latter may be overcrowded and thus there is a risk of running into unwanted traffic jams and congesting traffic even more. In particular we pass optimalRoad and Roads to perform a check on the Roads array and avoid returning the current optimalRoad that we discarded;
- the function $modifyRA()$ changes the percentage of availability of the selected road.

We will now provide the pseudo-code for the algorithm in Algorithm 1. As can be seen, it also makes use of four variables here described:

- **optimalRoad**: reports the current optimal road;
- **flag**: indicates whether the optimal road has been assigned to the user;
- **maxAvailability** : reports the percentage of the road with the best R.A. [4];
- **roadAvailability**: reports the percentage of the road. [4];
- **UserWeight**: indicates in percentage terms how much a vehicle affects road availability (depends on the vehicle used).

Finally, in the case where the user is assigned to a road, the algorithm will recalculate the percentage of the road available to him by reducing the assigned unit by the variable ”roadAvailability”. Having come to this point, the algorithm will return the road assigned to the user.

5 Benefits

We believe that the objectives in the design of a system, of a technology, in any field of application, should not focus purely on obtaining the desired functionality, but rather on obtaining the best possible optimisation. Until now, we have focused on offering the population **functional** solutions to answer the question: ”How do I get there?”. With this project, we tried to ask a different question: ”What is *the best way* to reach that place?”. Optimisation is now the basis of all modern systems and it would be absurd to imagine having an operating system that allocates resources to processes without implementing any kind of prioritisation strategy. In the same way, we believe that it is unthinkable not to concern ourselves with developing an algorithm that, based on real data in the short, medium and long term, allows all users who make a request to take the best route to their desired destination. We would therefore like to emphasise that our algorithm will provide a completely new

approach to this type of problem, dealing not only with providing a functionality, such as providing a link between two points, but rather with providing the best functionality and providing it not only to a small portion of users, but to everyone.

Strongly convinced of the importance of what we said above, let us now take into consideration and show you the differences between for example, an algorithm witch decides the appropriate path for a vehicle by just randomly assigning it, and our approach based on the banker's algorithm.

Algorithm 2 Random assignment of roads

Input: The data structures containing $Roads$ and $Users$
Output: The structure $assignments$ mapping a road to each user

1: **for** $user \in Users$ **do**
2: $assignment \leftarrow (user, RandomRoad(Roads))$
3: **end for**
4: **return** $assignment$

The *RandomRoad* function at line [2] assigns each user a road without performing any checks on it. In particular, by not checking the R.A. (Section 4) it also assigns users very congested roads, prolonging their travel time. Since there are no controls of any kind, gridlock situations and thus the creation of traffic jams are very frequent. The random algorithm does not distribute users equally on the roads at its disposal, but relies on randomness. For example, out of ten requesting users and with four roads available, it might assign all ten users to the same road.

On the other hand, our algorithm, taking into account the current availability of each road (R.A.), the percentage of manageable road (M.M.R.C.) and the maximum limit of assignable resource (M.A.R.), manages to ensure a correct distribution of requests on the roads at its disposal. This allows us to minimise the formation of traffic jams and, above all, to always provide the user with the most efficient route to the desired destination.

6 Conclusions

VANETs are among the most studied and used topics in many research fields, due to their potential in terms of self-organising and decentralising capabilities. One of the main and most interesting applications is traffic and road congestion detection, on which our research then focuses. By studying the data provided by these networks, and through the the banker's algorithm, we can decide how to assign each user to a route and distribute them within the road network at our disposal, by minimising the traffic congestion. From our point of view, it is important to emphasise that the integration between the VANET networks and the banker's algorithm is a highly interesting and innovative combo for the management of data flows concerning road networks and their relative traffic volume. The banker's algorithm has also been

used for practical application of temporal logics, such as in [13]. This gives us hints for future application of such a strong algorithm also on structures for knowledge modeling, by following the lead of what has been done in [12, 11].

References

1. Flora Amato, Valentina Casola, Andrea Gaglione, and Antonino Mazzeo. A semantic enriched data model for sensor network interoperability. *Simulation Modelling Practice and Theory*, 19(8):1745–1757, 2011.
2. Flora Amato, Luigi Coppolino, Francesco Mercaldo, Francesco Moscato, Roberto Nardone, and Antonella Santone. Can-bus attack detection with deep learning. *IEEE Transactions on Intelligent Transportation Systems*, 22(8):5081–5090, 2021.
3. Flora Amato, Nicola Mazzocca, and Francesco Moscato. Model driven design and evaluation of security level in orchestrated cloud services. *Journal of Network and Computer Applications*, 106:78–89, 2018.
4. Walter Balzano, Aniello Murano, Loredana Sorrentino, and Silvia Stranieri. Behavioral clustering: A new approach for traffic congestion evaluation. In *Workshops of the International Conference on Advanced Information Networking and Applications*, pages 1418–1427. Springer, 2020.
5. Hannes Hartenstein and LP Laberteaux. A tutorial survey on vehicular ad hoc networks. *IEEE Communications magazine*, 46(6):164–171, 2008.
6. S-D Lang. An extended banker's algorithm for deadlock avoidance. *IEEE Transactions on software engineering*, 25(3):428–432, 1999.
7. Marco Lapegna and Silvia Stranieri. Dclu: a direction-based clustering algorithm for vanets management. In *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 253–262. Springer, 2021.
8. Mark Lawley, Spyros Reveliotis, and Placid Ferreira. The application and evaluation of banker's algorithm for deadlock-free buffer space allocation in flexible manufacturing systems. *International Journal of Flexible Manufacturing Systems*, 10(1):73–100, 1998.
9. Youming Li. A modified banker's algorithm. In *Innovations and Advances in Computer, Information, Systems Sciences, and Engineering*, pages 277–281. Springer, 2013.
10. Hari Madduri and Raphael Finkel. Extension of the banker's algorithm for resource allocation in a distributed operating system. *Information Processing Letters*, 19(1):1–8, 1984.
11. Bastien Maubert, Aniello Murano, Sophie Pinchinat, François Schwarzentruber, and Silvia Stranieri. Dynamic epistemic logic games with epistemic temporal goals. In *ECAI 2020 - 24th European Conference on Artificial Intelligence*, volume 325, pages 155–162. IOS Press, 2020.
12. Bastien Maubert, Sophie Pinchinat, François Schwarzentruber, and Silvia Stranieri. Concurrent games in dynamic epistemic logic. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 1877–1883, 2020.
13. Johannes Francois Oberholzer et al. *Agent Interval Temporal Logic*. PhD thesis, University of Pretoria, 2020.
14. Muhammad Sameer Sheikh and Jun Liang. A comprehensive survey on vanet security services in traffic management system. *Wireless Communications and Mobile Computing*, 2019, 2019.
15. Statista. The cities with the worst traffic congestion, 2021.
16. Statista. Proportion of population in cities worldwide from 1985 to 2050, 2022.
17. A. Virag and S. Bogdan. Resource allocation in smart homes based on banker's algorithm. In *2011 2nd IEEE PES International Conference and Exhibition on Innovative Smart Grid Technologies*, pages 1–7, 2011.