Dynamic p-graphs for predictions in vehicular networks

Guillaume Béduneau, Ghada Jaber, Bertrand Ducourthial
Sorbonne Universités, Université de Technologie de Compiègne
CNRS UMR 7253 HEUDIASYC
CS 60319 - 57, avenue de Landshut, 60203 Compiègne, France
{guillaume.béduneau, ghada.jaber, bertrand.ducourthial}@hds.utc.fr

***Abstract*—Vehicular cooperation aims to solve traffic issues and to improve road safety for vehicles and pedestrians. Such a cooperation is done through distributed algorithms that have to be executed in a topology-changing vehicular network. Ensuring the success of the cooperative strategies is of particular importance for ITS. We propose a method to predict the performances of such distributed algorithms regarding the traffic conditions.**

***Index Terms*—vehicular network, dynamic graphs, performance prediction**

I. INTRODUCTION

Intelligent Transport Systems (ITS) that intend to solve road safety issues and traffic jams have been an important field of research in recent years. ITS are expected to include cooperation features (C-ITS) [1] that require vehicles to communicate with each other and exchange data such as perceptions and intentions. This is achieved by distributed algorithms that are implemented in the vehicles and in the infrastructure devices.

For legal reasons, it would be preferable that ITS services become predictable. Indeed, when an accident occurs, the software designer and the transportation system manager could be considered liable if they do not provide software efficiency guarantees. Moreover, the important ITS deployment costs (software design, hardware purchase and exploitation costs) urge the policy makers to make sure that they are improving road safety or road traffic fluidity. Therefore, the efficiency of the vehicular cooperation algorithms should be somehow guaranteed especially when used for safety applications.

However, the dynamic nature of vehicles can not ensure any distributed algorithm to succeed for every dynamic vehicular scenario. It remains always possible, using a scenario with faster movements to prevent some vehicles from exchanging enough information to achieve their cooperation. Therefore, any success prediction depends on the network dynamic.

To overcome the difficulties, we propose a method able to predict the performances of a distributed algorithm in a given road scenario. Starting from real road observations, the proposed method computes a modeling of the dynamic (namely *family of dynamic p-graphs*), which in turns is used to check some properties ensuring the success of the algorithm for a given use-case. We apply such a method to predict the behavior of some ITS services.

The paper is organized as follows. In section II, we present related works. Section III explains the prediction methodology. Section IV describes experiments intended to validate the accuracy of the prediction and their results. Finally, Section V concludes the paper with some perspectives.

II. RELATED WORK

Dynamic networks gain interest for their applications related to communication in mobile systems, satellites constellation, vehicles cooperation, fleets of robots... A lot of works are conducted about them but predicting a distributed algorithm behavior in a vehicular network remains complex. This section briefly reviews the existing models for dynamic networks and their use for performance prediction in vehicular networks.

A. Static graphs sequences

As static networks are usually modelled with graphs, dynamic networks were first modeled with adapted graphs taking into account the time evolution. In [3], dynamic networks are represented with a sequence of graphs, in which graphs could be studied independantly [4] as static ones. While this method is simple to use (graphs are well known models with a lot of interesting properties), it may be less useful if the studied period goes through a transition because graph properties do not take transitions into account. When transitions occur more often, they can not be ignored and the model should represent temporal evolution within the graph.

B. Dynamic graphs

To be able to study graphs through transitions, it is possible to consider and study the whole sequence of graphs, with its own properties. The notion of *evolving graphs* is proposed in [5]. It takes into account time evolution while proposing interesting temporal properties such as the *journey* which is a path that respects temporal network topology evolution. The evolving graph may be timed to consider different edge durations.

In [6], the authors propose a formalism called *TVG (for Time Varying Graphs)* in which any dynamic network may be fully

---

This work was carried out and funded in the framework of the Labex MS2T. It was supported by the French Government, through the program "Investments for the future" managed by the *Agence Nationale de la Recherche* (Reference ANR-11-IDEX-0004-02) and through the *Agence Innovation Défense*.

described. TVGs are classified by their dynamic properties [11] and adaptations are made to static graph properties to cope with dynamic topologies [?]. This classification may help algorithm design and analysis [10]. The TVG formalism includes a *duration function*, that can be used for journey duration calculation. This latter performs well for Delay Tolerant Networks (DTN) analysis [7] and epidemiology [8], [9] where nodes are numerous and end-to-end communication may have a high duration.

For vehicular networks, the cooperation algorithms depend more on the other nodes and latency has to be very low. Moreover, the cooperation usually takes place within a small group of nearby vehicles.

For this purpose dynamic p-graphs have been proposed in [12] with the particularity to take into account the network capacities of edges directly in the model. This feature allows to drop edges that would appear in a TVG without being usable by the communication protocol. It appears to be suitable for vehicular cooperation algorithms, where nodes have to exchange several messages in a close neighborhood, generally with low end-to-end communication duration.

C. Performances prediction

While designing a distributed algorithm intended to be used in a vehicular context, designer usually test its performances under few specific traffic scenarios (often in a simulator). This validates the algorithm and gives an idea of its performances but there is no guarantee that the performances with another scenario will be similar to the explored one. This may limit the real deployments of such algorithms or force the algorithm designer to make tests with a very large number of scenarios (potentially infinite) which come with costs. Moreover, each algorithm update would lead to the same situation as creating a new one.

An ITS manager would like to know, when a vehicular cooperation is proposed, what performances it should expect in its specific conditions before engaging any expensive implementation or deployment. Though road traffic prediction are common for the drivers, it seems there is no solution in the literature for predicting the behavior of cooperative algorithms.

To fill the gap between simulation and deployment, we propose a method for predicting the behavior of distributed systems. It is based on so-called dynamic p-graphs, which proposes a modeling of the dynamics that encompasses both the nodes mobility and the communication protocol efficiency. This method has the ability to increase the reproducibility of the studies related to vehicular cooperation. Moreover, it reveals to be useful for safety critical studies when it becomes important to ensure some performances.

III. PREDICTION METHODOLOGY

This section presents the proposed method to predict the performances of various distributed algorithms intended to be used in vehicular networks. We first present the p-graphs modeling it relies on before explaining the performance study. Then, we describe the prediction method and the used tools.

A. Family of dynamic p-graphs

A static network is usually represented by a graph $G(V, E)$ where vertices of V represent the nodes and the edges of E linking two nodes represent their capacity to communicate with each others. However, for dynamic networks, the evolution of the topology gives a succession of static timed graphs (see Section II) called in the following *observation*.

Starting from such an observation, the dynamic p-edges are determined. A *dynamic p-edge* (u, v) exists whenever its duration is sufficient for the communication of p messages [12]. A dynamic p-edge depends both on the nodes mobility and on the communication protocol properties. Indeed, if the node speed increases then less messages would be exchanged before leaving the communication range. In case the throughput or the range of the communication protocol increases, more messages could be exchanged.

If (u, v) is a dynamic p-edge, then it is also a dynamic q-edge for every integer q satisfying $0 < q < p$. The *maximal order* of a p-edge is the number of messages it can transport before disappearing. Edges with an infinite duration do not admit a maximal order.

Considering all p-edges for a given value of p define a sequence of graphs called *dynamic p-graphs*. By starting with $p = 1$ and incrementing p, it is possible to get a set of dynamic p-graphs encompassing both the nodes mobility and the communication protocol. Considering that the observation edges have either a finite bounded duration or last forever, it is possible to prove that, after a certain value of p, every dynamic p-graph representing the scenario are identical, meaning that the further values of p may be ignored. Hence, any vehicular scenario leads to a unique finite set of dynamic p-graphs [12], called a *family* of dynamic p-graphs. The dynamic p-graph family faithfully represents the dynamics of the scenario for our purpose while being finite.

B. Algorithm study

Distributed algorithms are defined using specification which may be based on liveness and safety properties for instance.

To compute a prediction related to a distributed algorithm running in a given scenario, it is possible to convert the algorithm specification into a property that depends on conditions over the dynamic p-graphs family representing the scenario.

This specification expression can be refined considering specific use-cases where a performance has to be reached. For instance, a diffusion algorithm that is used for alert transmission should have a low latency while the same diffusion algorithm should have a sufficient flowrate if it diffuses high definition map fragments. Such *performance properties* could be combined in order to obtain multi-criteria study.

Let take an example. Consider the algorithm sending a fragmented piece of data from one node to another. It would fulfill its success property if "The destination node gets the whole data during the scenario". If the use case is an RSU (Road Side Unit) sending a fragmented map (composed of 8 fragments) to the vehicles on the road by using DSRC (Dedicated Short Range Communications) communication, the performance property could be “Any vehicle that remains on the road and whose speed remains below the speed limit gets the whole data before leaving the RSU range”.

Note that neither the algorithm specification expression nor the performance properties have necessarily to be computed in real time because they are only used for the algorithm behavior prevision.

The performance properties is easily expressed using dynamic p-graphs. For instance, an example of a performance property for the previous example could be the existence of a dynamic 8-edge linking the RSU and every vehicle.

C. Prediction

Once a performance property has been expressed, the prediction can properly take place. The observation can be made from a trajectory history of all involved nodes: for every node, a link connects it to others depending on the communication range. The dynamic p-graphs family is then computed, taking into account the communication protocol. The performance property is looked for in the dynamic p-graph family and enables to conclude whether the algorithm would succeed in the tested scenario.

The only experiment such a prediction requires is getting real observation. It could have taken place even before the conception of the tested algorithm and do even not need the vehicles to have been able to communicate. This can be useful to size the communication protocol, or the scenarios parameters (before implementing real conditions.

The most difficult step of the prediction methodology is the algorithm study that may not yet be automatable. This step is only done once and can be used on a variety of scenarios.

Figure 1 presents the method steps for an algorithm and a given observation.

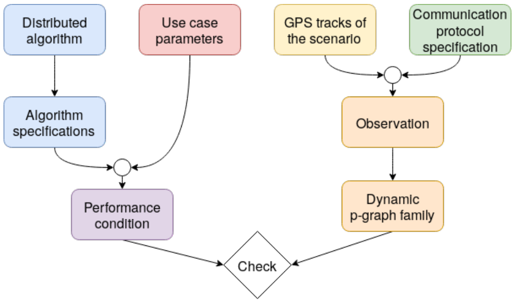

Fig. 1. Summary of the proposed prediction method

D. Toolchain

The implementation of the method requires several tools. In the real world, getting an observation remains a bit complex and costly. For practical reasons, we use a dynamic network emulator, belonging to the Airplug middleware that enables to use the exact same implementation of distributed algorithm for emulation and real condition tests. The emulated conditions have been shown to be conform with real world experiments [13]. The emulator can directly use real GPS tracks to produce the observation needed to model the scenario with a dynamic p-graph family for various scenarios.

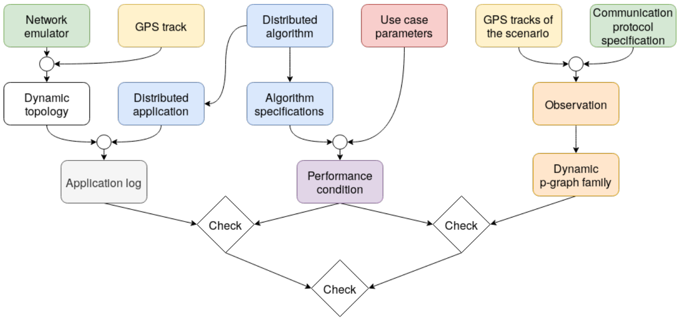

Fig. 2. Summary of the experimental protocol used to validate the prediction method

To compute the dynamic p-graph family, a program named pgraphtool takes as input the observation and the communication parameters (duration to transmit 1 message) and has as output the dynamic p-graph family describing the scenario. Regular expressions allow checking whether the dynamic p-graph family satisfies the performance property.

IV. EXPERIMENTS

This section presents the validation of our method by means of experiments illustrated with an easy to understand example and its results. We then describe the design, the use cases, the algorithm and the expression of performance property.

A. Overview

To validate the prediction methodology, a ground truth is necessary: comparing the prediction and ground truth results in the *prediction accuracy*.

This ground truth is obtained through the execution of the distributed application in the prediction scenario. For this purpose, we use the network emulator introduced previously. The application generates log files containing its state at different moments of the execution. Tiny scripts are used on the log files to check whether the application did fulfill the success condition or not.

Independently, a prediction is made on the same scenario using the methodology presented in the previous section.

The ground truth is used to validate the prediction predicate, representing the success (predicted or real) of the execution. The prediction accuracy can either be:

- a predicted algorithmic success
- a predicted algorithmic failure
- an unpredicted algorithmic success
- an unpredicted algorithmic failure

Experiments with 100% of algorithmic success or 100% of algorithmic failures could result from selecting too permissive or restrictive performance properties. Indeed, setting infinitely harsh conditions would make easy to predict for any algorithm that it would fail. On the contrary, too favorable conditions would allow easily predicted algorithmic successes. The validation experiments will therefore take place in scenarios allowing both ends.

Finally, as our method intends to be usable for any cooperative algorithm, its validation requires to test a variety of such algorithms.

B. Experiment setup

By lack of place, a single algorithm is presented here; it has been chosen for its pedagogical interest. The algorithm to be tested is a beaconing algorithm called *Beacon*: every vehicle sends periodic beacons including its own identifier and stores the received messages.

The algorithm is tested with a variety of scenarios: a car is overtaking a truck (a slower vehicle whose speed is around 90 km/h) on the highway, after having been in the range of an RSU. The car may get several more fragments from the truck re-transmissions if its speed prevented full reception. The car speed may be either 110, 120 or 130 km/h.

With this algorithm, the number of beacons received by the car is obtained by adding the capacities of the different edges between the truck and the car. The algorithm is considered successful if the vehicle received at least 20 beacons. The vehicles start randomly located on their trajectory so that the car has to overtake the truck.

C. Results

For each experiment, we get either a failure or a success of the algorithm. We use bar charts for presenting them in Figure 3. We observe a very good prediction.

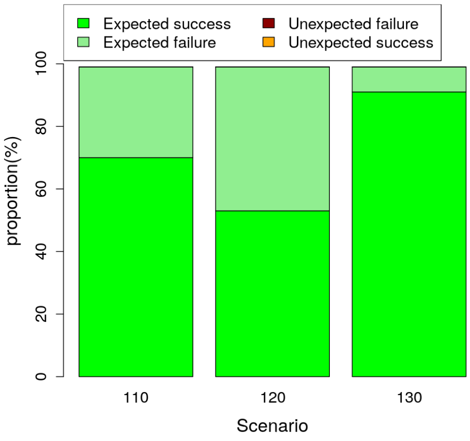

Fig. 3. Prediction accuracy for Beacon algorithm: scenario is a car overtaking a truck on a highway, with various speeds (110, 120, and 130 km/h).

Considering the algorithm is simple and the scenario only involves two nodes, the prediction accuracy is very good for both success and failure predictions. These results even show there was no prediction errors which validates the proposed method. Very good predictions have also been observed for more complex algorithms and use-cases; they cannot be presented here by lack of place.

V. CONCLUSION

In this paper, we presented a method for predicting the behavior of cooperative algorithms in dynamic networks. Starting from the observation of real scenarios, a complete modeling of the network dynamic is obtained through so-called p-dynamic graphs. Such a modeling is convenient for expressing algorithmic properties characterizing their success and/or performance. A simple tool chain then allows checking such conditions without having to run long and complex simulations. Our prediction method has been validated through many experiments (a single one has been presented here as illustration) and the results are promising. Such an approach could allow giving safety guarantee before deploying a distributed vehicular cooperation algorithm; it could also help to identify what to improve in order to reach better algorithmic performance in a given deployment.

Though the method requires the formulation of algorithmic properties, our examples show that this is not a serious drawback. But this point will be confirmed in the future when studying more cases. Another future work consists in the integration of the packet losses directly into the modeling.

REFERENCES

[1] "A European strategy on Cooperative Intelligent Transport Systems, a milestone towards cooperative, connected and automated mobility", European Commission.

[2] A. Matthias, "The responsibility gap: Ascribing responsibility for the actions of learning automata," Ethics and information technology, 6(3), 175-183, 2004.

[3] G. N. Frederickson, "Data structures for on-line updating of minimum spanning trees, with applications," In SIAM Journal on Computing, 14(4), 781-798, 1985.

[4] J. H. Reif, "A topological approach to dynamic graph connectivity," Information Processing Letters, 25(1), 65-70, 1987.

[5] B.-M. Bui-Xuan, A. Ferreira, and A. Jarry, "Computing shortest, fastest, and foremost journeys in dynamic networks," In International Journal of Foundations of Computer Science, 14(2):267285, 2003.

[6] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro, "Time-varying graphs and dynamic networks," In International Journal of Parallel, Emergent and Distributed Systems, 27(5):387408, 2012.

[7] A. Casteigts, P. Flocchini, B. Mans, and N. Santoro, "Measuring temporal lags in delay-tolerant networks," IEEE Transactions on Computers, 63(2), 397-410, 2012.

[8] J. Enright, K. Meeks,G. B. Mertzios, and V. Zamaraev, "Deleting Edges to Restrict the Size of an Epidemic in Temporal Networks," In 44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik,2019.

[9] G. B. Mertzios, H. Molter, R. Niedermeier, V.Zamaraev, and P. Zschoche, "Computing Maximum Matchings in Temporal Graphs," In STACS (Vol. 154, p. 27), Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[10] Altisen, Karine, et al. "Self-stabilizing systems in spite of high dynamics." International Conference on Distributed Computing and Networking, 2021.

[11] Casteigts, Arnaud, et al. "Robustness: A new form of heredity motivated by dynamic networks." Theoretical Computer Science (vol 806, p. 429), 2020.

[12] B. Ducourthial, and A. Wade, "Dynamic p-graphs for capturing the dynamics of distributed systems." Ad Hoc Networks, 50, 13-22, 2016.

[13] A. Buisset, B. Ducourthial, F. El Ali, and S. Khalfallah, "Vehicular networks emulation," In 2010 Proceedings of 19th International Conference on Computer Communications and Networks (pp. 1-7). IEEE, 2010.

[14] A. Segall, "Distributed network protocols," IEEE Transactions on Information Theory, 1983.