Báguena Albaladejo, M.; Tornell, SM.; Torres Cortes, A.; Tavares De Araujo Cesariny Calafate, CM.; Cano Escribá, JC.; Manzoni, P. (2013). VACaMobil: VANET Car Mobility Manager for OMNeT++. 3rd IEEE International Workshop on Smart Communication Protocols and Algorithms (SCPA 2013) in conjunction with IEEE ICC'13. IEEE. doi:10.1109/ICCW.2013.6649393.

VACaMobil: VANET Car Mobility Manager for OMNeT++

Miguel Báguena, Sergio M. Tornell, Álvaro Torres, Carlos T. Calafate, Juan-Carlos Cano, Pietro Manzoni
Department of Computer Engineering,
Universitat Politècnica de València
Camino de Vera S/N 46022, Valencia, Spain.
mibaal@upvnet.upv.es, sermarto@upv.es, atcortes@batousay.com, {calafate, jucano, pmanzoni}@disca.upv.es

***Abstract*—The performance of communication protocols in vehicular networks highly depends on the mobility pattern. Therefore, one of the most important issues when simulating this kind of protocols is how to properly model vehicular mobility. In this paper we present VACaMobil, a VANET Car Mobility Manager for the OMNeT++ simulator which allows researchers to completely define vehicular mobility by setting the desired average number of vehicles along with its upper and lower bounds. We compare VACaMobil against other common methods employed to generate vehicular mobility. Results clearly show the advantages of the VACaMobil tool when distributing vehicles in a real scenario, becoming one of the best mobility generators to evaluate the performance of different communication protocols and algorithms in VANET environments.**

***Index Terms*—Vehicular Networks, Mobility patterns, Simulation Tool, SUMO, TraCI.**

I. INTRODUCTION

The reproducibility of experiments is a major issue when evaluating smart communication protocols and algorithms, especially over Vehicular Ad-hoc NETworks (VANETs). In [6] the authors provide a complete review of the minimum set of parameters that should be identified in order to allow other researchers to reproduce simulation experiments. They pointed out several key parameters, such as the simulated hardware, the network simulator, the scenario, and the road traffic simulator. However, regarding node mobility, there are other parameters that have been mostly ignored by the research community: the density of traffic and the traffic demand.

As other authors pointed out in previous studies, mobility models [10] and the chosen scenario [4], as well as the node density, heavily influence the final network performance. However, since mobility generators and road traffic simulators are often tough to configure, the simulated node density and distribution may depend on complex data that are usually not included in the published academic results, thereby compromising reproducibility.

In this paper we present VACaMobil (VANET Car Mobility manager), a mobility manager module for the OMNeT++ simulator which is the first, to the best of our knowledge, able to generate SUMO [1] driven nodes in a vehicular network while ensuring certain user-defined parameters, such as the average, maximum, and minimum number of vehicles. This goal is useful for mid-length simulations, typically one hour, allowing researchers to assume that the vehicle density is stable. At the same time, since our solution is tightly coupled with SUMO through the TraCI interface, it is able to mimic real vehicle behavior. By running in parallel with SUMO, VACaMobil executes the following tasks: (i) it manages when a new vehicle must be introduced in the network, (ii) it assigns a random route from a predefined set to each vehicle, and (iii) it determines which type of vehicle should be added. When using VACaMobil, and given a specific road map, researchers will be able to completely define the network mobility merely by defining the desired average number of vehicles and its standard deviation value (upper and lower bounds).

Going a step further, our tool also aids researchers at selecting among the different types of vehicles previously defined in SUMO, such as “cars”, “buses”, or “trucks”. This allows researchers to easily define road traffic simulations with heterogeneous vehicles.

The rest of this paper is organized as follows: In section II, we shortly introduce the different methods for generating VANET mobility patterns that the research community commonly uses. In section III, VACaMobil is fully described. In section IV, we compare our proposal with the duarouter and duaIterate.py tools, both included in SUMO. Finally, in section V, we present our conclusions and some future plans to improve VACaMobil.

II. A REVIEW OF EXISTING MOBILITY GENERATORS FOR VANETS

Before presenting the details of our proposal, we analyze some of the methods commonly used to obtain suitable mobility patterns in urban vehicular scenarios. We have analyzed several papers published during the last few years, most of them published in conferences and journals related to Intelligent Transportation Systems. Early approaches relied on too simple mobility models based merely on random mobility. Since these simple models do not represent vehicle mobility properly, other mobility models have been recently developed based on real world traces and also on artificial mobility models from the field of transportation and traffic science. In this section, we briefly describe the most relevant works.

A. Random Vehicle Movement

At the beginning of the previous decade, the “Random Way-Point" was extensively used in Mobility Ad-Hoc NETwork

(MANET) research. However, in 2003, the authors in [15] demonstrated how harmful the Random Way-Point mobility model really is. Moreover, the effects described in this work are even worse when simulating VANETs. Later on, some other authors have extended the "Random Way-Point" mobility model by restricting the mobility of nodes to a map layout, as in [14]. However, this improvement does not solve the majority of the "Random Way-Point" model problems stated previously.

In our research group we developed a tool called "CityMob" [9]. CityMob allows users to create random mobility patters restricted to a grid. It also adds support for *downtown* definition, where a *downtown* is a region inside the simulated map which concentrates the majority of the selected routes along the simulation. Although CityMob presents a big improvement compared to non restricted mobility models, as well as random mobility models, it also presents some problems; the most important one is that vehicular mobility is not influenced by other vehicles, *i.e.* two different vehicles can occupy the same location and no minimal distance between vehicles is required. Moreover, vehicles do not change their speed during a trip. However, in the real world, vehicles continuously change their speed according to traffic conditions and road characteristics. Last but not least, vehicles keep moving throughout the whole simulation, which especially influences the performance of protocols that keep data stored in buffers. The research community quickly realized the problems derived from inaccurate simulation patterns, and started to work in other methods to obtain suitable mobility traces.

B. Real Mobility Traces

Compared to the use of random mobility, real traces present a clear improvement. Such traces are usually obtained from a certain set of nodes, e.g. from taxis in the city of Shangai [7]. Mobility traces can be obtained by tracking the mobility of nodes using On-Board units, as in [7], or by using road-side equipment, as in [5]. Although real traces represent the most realistic mobility patterns, we can not obviate the fact that the mobility of the tracked nodes is highly influenced by the movement of other non tracked vehicles, *e.g.* taxis' mobility is influenced by other users on the road whose movement is not reflected in the collected traces. Moreover, real traces lack the flexibility to allow for an exhaustive evaluation of VANET protocols, *e.g.* changing the vehicle density without modifying their speed is clearly unreal.

C. Assisted Traffic Simulation

The restrictions of real traces can be overcome, with almost no loss of realism, by using mobility models taken from the field of transportation and traffic science. Several road traffic simulators are widely used among the VANET research community. One of the most widely used mobility generators is SUMO [1]. When simulating traffic mobility for VANETs not only the vehicles' behavior is important, but also the traffic demand. SUMO allows defining traffic demand in two different ways: trips and flows. The former defines only a vehicle, its origin and its destination, while the latter defines a set of vehicles which execute the same trip. SUMO currently provides several tools to generate traffic demand:

- randomTrips.py: A random trip generator. This tool generates a trip every second having a random origin and destination. It does not check if the origin and destination are connected, or whether the trip is possible.
- duarouter: A Dijkstra router. Given a file with trips and flows, this tool generates the actual traffic demand, expressed in vehicles with an assigned route. Routes are calculated using the Dijkstra algorithm, and every unconnected trip is discarded.
- duaIterate.py: This python script will produce a set of optimal routes from a trip file, *i.e.* all the nodes will follow that route which minimizes the total trip-time for all nodes. This tool repeats a routing-simulation loop until optimal routes are found.

Authors have used these tools in order to generate traffic demands for SUMO. The most simplistic one is to define different flows inside the network. Although drivers usually move from certain districts to others, following patterns associated with their working and living places, defining the traffic only by creating fixed flows lacks of any realism, as we can see in [2] where only a few flows are defined by the user. Another common approach is to generate random trips using *randomTrips.py*. This approach presents the problem that only one vehicle is introduced every second, which leads to long transitory periods until the network reaches a stable state. A more sophisticated traffic demand generation strategy is presented in [8], where a predefined number of vehicles following random routes are randomly placed at the beginning of the simulation. Following this trend, in previous works we used C4R [3], which is a software developed by our group to automate the task of generating random vehicles with random routes at random places. The work presented in [13] is the only one that we could find which uses the *duaIterate.py* script to generate a "stable and optimal distribution of flows". This type of traffic definition presents a problem: the trip duration can not be predicted before running the simulations, and, as a consequence, there is no way to ensure, or even determine, if the road traffic simulation will last until the end of the network simulation. As some works have stated before, this lack of realism and generality in mobility patterns can lead to biased results [10].

D. Bidirectionally coupled network and traffic simulations

In [11] its authors go a step further and present a new simulation framework called Veins, which includes the TraCI interface to allow the network simulator to interact with the traffic simulator running in parallel. Although, it presents much novelty and opens a lot of possibilities for VANET simulation, authors do not address the traffic demand generation problem. This framework demonstrated its new characteristics in [12], and it is one of the main elements of our VACaMobil module, allowing us to interact with SUMO during the network simulation and create new vehicles.

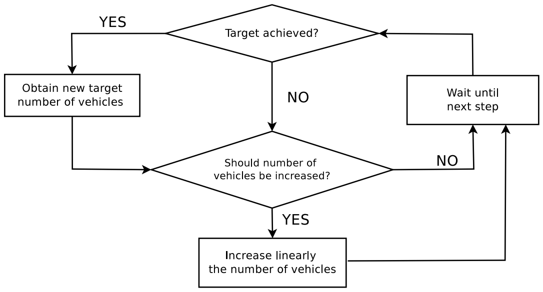

Figure 1. Main loop of the VACaMobil tool.

III. VACAMOBIL MOBILITY MANAGER

In this section we present our VANET mobility generator providing the main characteristics and its implementation details.

A. Characteristics

The main characteristic of VACaMobil is to offer realistic mobility scenarios. To that end it can guarantee an average number of vehicles while keeping the current number of vehicles within the given upper and lower bounds, the latter being associated to the defined standard deviation value. These features allowed creating a tool that ensures repeatability of network simulations under the same road traffic conditions just by defining the average number of vehicles and the standard deviation value.

Another characteristic it offers is the possibility of introducing types of vehicles in simulations, such as "car", "bus" and "truck", each one with its own characteristics. This is an important feature because high level decisions may be based on the type of the vehicle.

VACaMobil also provides a realistic vehicle distribution based on a list of different predefined routes.

Finally it works on-line with the SUMO traffic simulator, obtaining all the needed information about routes and type of vehicles through the TraCI communication interface, thereby avoiding the duplicity of configuration files.

B. Implementation details

This tool extends the module collection available in the Veins framework [11] with new capabilities. We explain, for each of the characteristics described before, the different implementation decisions taken.

1) Average number of vehicles: Figure 1 shows the VACaMobil iterative control loop. At every step of the mobility simulation, VACaMobil compares the current number of vehicles in the simulation with the target number of vehicles. Depending on whether it is greater or lower than the target value, VACaMobil waits until the number of vehicles decreases towards the target value, or, in the second case, starts inserting several new vehicles in every step of the mobility simulation in order to increase the current value until the desired value is achieved. To avoid having an average number of vehicles higher than the one defined by the user, the time during which new vehicles are inserted is as long as the last period where the number of vehicles has decreased. By following this approach we also avoid high frequency fluctuations in the total number of vehicles throughout the simulation time.

Values for the *target number of vehicles* variable are obtained from a normal distribution whose mean is the desired average number of vehicles, and whose standard deviation is defined by the user. Its value is limited to the upper and lower bounds, which are defined as the $mean \pm 3 * standard\,deviation$: this value avoids extremely high or extremely low values for the current number of vehicles.

2) Different types of vehicles: One of the parameters we can obtain via TraCI indicates the types of vehicles that are available in the traffic simulation. The user can set different probabilities associated to each vehicle type. In this case, every time a new car is generated, we obtain a uniform random value and select the correspondent vehicle type. If no probability is defined for a certain type of vehicles, we assume it is equal to 0. However, if no probability value is assigned to any of the defined types of vehicle, only vehicles of the first defined type will be generated.

3) Routing set and vehicle distribution: Since SUMO itself loads all the different routes at startup, we can also retrieve them through TraCI, and, as in the previous item, we select one of them with an uniform probability every time we generate a new vehicle.

VACaMobil does not compute routes at simulation time; instead it relies on the goodness of the different routes made available by SUMO. To guarantee that vehicles are distributed realistically, we also developed a tool based on *duaIterate.py* and *randomTrips.py* which creates a SUMO route file with several random routes.

Finally, to ensure that a new vehicle is correctly added, the default behavior is to attempt to insert the vehicle at any of the lanes available on the first edge of the route. If the edge is full, the module selects a new route sequentially from a list, repeating the operation until it finds a free place to insert the vehicle, or until the first selected route is selected again, which means that there is no room on the road map for the new vehicle.

IV. EVALUATION

In this section we evaluate VACaMobil to verify whether the objectives and characteristics described in section III-A have been accomplished. In order to do that, we have compared VACaMobil against the tools currently included in SUMO, *i.e.* duarouter and duaIterate.py, that were described in section II. We have selected the following scenarios:

- **Synthetic Manhattan scenario:** We created a road map consisting of a 25 x 25 grid with segments of 200 meters (Figure 2).
- **Urban real map scenario:** We extracted an urban road layout from the OpenStreetMap database. It is a scenario of about 7 km2 from the city of Milano characterized by short road segments and a high road density (Figure3).

In both scenarios, the set of random routes provided by VACaMobil is extracted from the traffic demand generated by duaIterate.py. In the following subsection, we compare the vehicle density and its evolution along the simulation time for the aforementioned tools and scenarios.

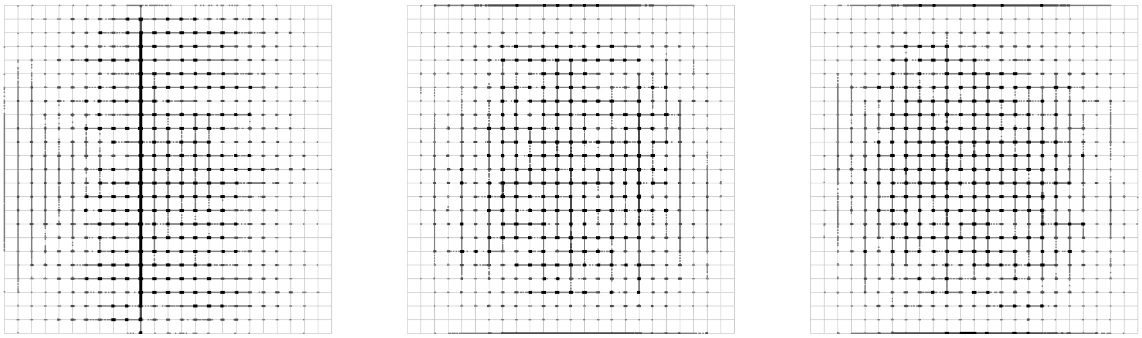

Figure 2. Heat map for the Manhattan scenario when using duarouter, *duaIterate.py*, and VACaMobil (from left to right).

Figure 3. Heat map for the urban scenario when using duarouter, *duaIterate.py*, and VACaMobil (from left to right).

A. *Vehicle distribution study*

We first evaluate one of the most important issues in vehicular mobility: how vehicles are distributed over the simulated road map.

Figure 2 shows how the compared methods perform in the Manhattan scenario. Due to its lack of randomness, duarouter is unable to select different routes for vehicles when there are several streets with the same travel-time. This prevents the simulator to properly distribute vehicles, and so all them are routed through the same street (eleventh from the left). When using the *duaIterate.py* script, a better distribution of the vehicles is achieved due to the fact that many simulations are sequentially executed to optimize vehicle routes. Since the VACaMobil's random routes set is obtained from duaIterate.py, it achieves a similar nodes distribution.

Figure 3 shows performance results for the urban scenario. In this case, duarouter is also unable to spread the vehicles properly. Since some roads are faster than others, all the vehicles are routed through them, even when these streets are congested. Therefore, an undesired traffic congestion is created in the fastest inner roads. However, this is an unrealistic scenario because drivers tend to avoid traffic jams whenever possible. When either using *duaIterate.py* or VACaMobil, vehicles are routed through alternative streets, avoiding traffic jams. This strategy has a higher degree of similitude compared to real road traffic, since drivers prefer faster roads but often change their route to avoid traffic jams.

B. *Vehicle density study*

To make simulations more easily comparable, a similar traffic density is desirable in all simulated city layouts. Current tools can not correctly handle this problem. In order to compare the behavior of the three selected methods previously presented, we have measured the average number of vehicles, its standard deviation, and its evolution along simulation time.

Table I
VACAMOBIL CONFIGURATION

	Vehicle number	Std. dev.
Manhattan	320	6
Urban scenario	370	8

Table II
VEHICLE STATISTICS SUMMARY

	Manhattan		Urban scenario	
	mean	std. dev.	mean	std. dev.
duarouter	313.767	58.8271	880.546	465.716
duaIterate.py	304.487	55.5174	393.717	96.414
VACaMobil	319.349	6.14267	369.691	7.84640

Table II shows the differences in terms of number of vehicles for the two target scenarios for each traffic generation tool. In the simplest scenarios (Manhattan), the three methods can achieve a stable value for the mean vehicle density with a low standard deviation. However, neither duarouter nor duaIterate allow to *a priori* configure the value of this parameter. On the contrary, VACaMobil is not only able to populate the network with the desired number of vehicles, but also allows defining a maximum and a minimum number of vehicles by using the standard deviation feature, which will bound the number of vehicles. In complex maps like the urban scenario, VACaMobil is the only tool able to maintain the standard deviation value within the predefined bounds.

To better understand the aforementioned values, figures 4, and 5 show the number of vehicles in the scenario along time for each tool. Since *duarouter* and *duaIterate.py* are only able to add one vehicle per second, the user cannot predict

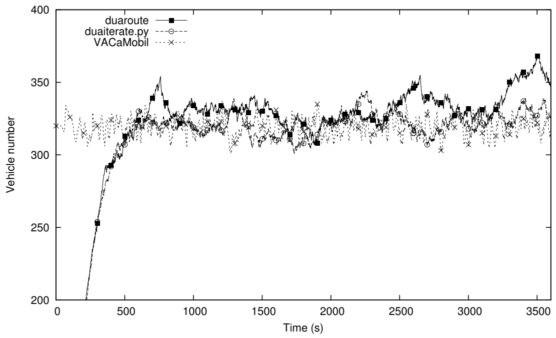

Figure 4. Vehicle number evolution for the Manhattan scenario.

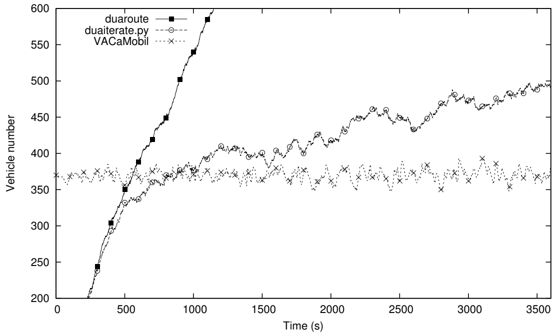

Figure 5. Vehicle number evolution for the urban scenario.

when vehicles will arrive to their destination and disappear from the network. Therefore, the number of vehicles when the simulation reaches the steady-state for the Manhattan scenario is not known *a priori*, converting protocol analysis based on number of vehicles in a mere act of faith. Moreover, in urban maps where traffic jams are very common, as in the urban scenario, it takes more time for vehicles to reach their destination and leave the network, which leads to a constantly increasing number of vehicles in the network when not using VACaMobil. Comparing the configuration in table I and the results in table II, we can conclude that both the target number of vehicles and the standard deviation goal are clearly achieved with our VACaMobil approach.

V. CONCLUSIONS AND FUTURE WORK

In this paper we presented VACaMobil[1], a new mobility manager for the OMNeT++ simulator which promotes the full repeatability of VANET simulations. In particular, we added critical features to previously existing tools, such as ensuring a constant number of vehicles during the entire simulation period, disseminating vehicles throughout the whole route-map, and offering the possibility of defining different vehicle types with different probabilities.

Contrarily to other existing tools, which are not able to control the mean number of vehicles nor its standard deviation, VACaMobil is able to maintain the mean number of vehicles and the standard deviation value within user-defined bounds. To the best of our knowledge, this is currently the only tool that allows studying a vehicular network in a steady situation without losing the realistic vehicle behavior provided by SUMO.

As future work we plan to improve VACaMobil offering downtown definition and automatic placement of Road Side Units (RSU).

ACKNOWLEDGEMENTS

This work was partially supported by the *Ministerio de Economía y Competitividad*, Spain, under Grants TIN2011-27543-C03-01 and BES-2012-052673, and by the *Ministerio de Educación*, Spain, under the FPU program, AP2010-4397, AP2009-2415.

REFERENCES

[1] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz. Sumo - simulation of urban mobility: An overview. In *SIMUL 2011, The Third International Conference on Advances in System Simulation*, pages 63–68, Barcelona, Spain, October 2011.

[2] L.-J. Chen, Y.-Y. Chen, K. chan Lan, and C.-M. Chou. Localized data dissemination in vehicular sensing networks. In *Vehicular Networking Conference (VNC), 2009 IEEE*, pages 1 –6, oct. 2009.

[3] M. Fogue, P. Garrido, F. J. Martinez, J.-C. Cano, C. T. Calafate, and P. Manzoni. Using roadmap profiling to enhance the warning message dissemination in vehicular environments. In *36th IEEE Conference on Local Computer Networks (LCN 2011)*, Bonn, Germany, October 2011.

[4] M. Fogue, P. Garrido, F. J. Martinez, J.-C. Cano, C. T. Calafate, and P. Manzoni. An adaptive system based on roadmap profiling to enhance warning message dissemination in vanets. *Networking, IEEE/ACM Transactions on*, PP(99):1, 2012.

[5] M. Gramaglia, M. Calderon, and C. Bernardos. Trebol: Tree-based routing and address autoconfiguration for vehicle-to-internet communications. In *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*, pages 1 –5, may 2011.

[6] S. Joerer, C. Sommer, and F. Dressler. Toward Reproducibility and Comparability of IVC Simulation Studies: A Literature Survey. *IEEE Communications Magazine*, 50(10):82–88, October 2012.

[7] X. Li, W. Shu, M. Li, H. Huang, and M.-Y. Wu. DTN Routing in Vehicular Sensor Networks. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pages 1–5. IEEE, Nov. 2008.

[8] C.-C. Lo, J.-W. Lee, C.-H. Lin, M.-F. Horng, and Y.-H. Kuo. A cooperative destination discovery scheme to support adaptive routing in vanets. In *Vehicular Networking Conference (VNC), 2010 IEEE*, pages 202 –208, dec. 2010.

[9] F. Martinez, J.-C. Cano, C. Calafate, and P. Manzoni. Citymob: A mobility model pattern generator for vanets. In *Communications Workshops, 2008. ICC Workshops '08. IEEE International Conference on*, pages 370 –374, may 2008.

[10] C. Sommer and F. Dressler. Progressing toward realistic mobility models in vanet simulations. *Communications Magazine, IEEE*, 46(11):132 – 137, november 2008.

[11] C. Sommer, R. German, and F. Dressler. Bidirectionally coupled network and road traffic simulation for improved ivc analysis. *Mobile Computing, IEEE Transactions on*, 10(1):3 –15, jan. 2011.

[12] C. Sommer, O. Tonguz, and F. Dressler. Adaptive beaconing for delay-sensitive and congestion-aware traffic information systems. In *Vehicular Networking Conference (VNC), 2010 IEEE*, pages 1 –8, dec. 2010.

[13] C. Sommer, O. Tonguz, and F. Dressler. Traffic information systems: efficient message dissemination via adaptive beaconing. *Communications Magazine, IEEE*, 49(5):173 –179, may 2011.

[14] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, WDTN '05, pages 252–259, New York, NY, USA, 2005. ACM.

[15] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 2, pages 1312 – 1321 vol.2, march-3 april 2003.

---

[1] VACaMobil is freely available at *www.grc.upv.es/software*.