View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Repository of the Academy's Library

2015 Models and Technologies for Intelligent Transportation Systems (MT-ITS)
3-5. June 2015. Budapest, Hungary

Simulation Environment for Investigating Crowd-sensing Based Urban Parking

Károly Farkas
Department of Networked Systems and Services
Budapest University of Technology and Economics
Budapest, Hungary
farkask@hit.bme.hu

Imre Lendák
Faculty of Technical Sciences
University of Novi Sad
Novi Sad, Serbia
lendak@uns.ac.rs

***Abstract*— This paper introduces a simulation environment developed for analyzing crowd-sensing based applications in the Smart City application domain. As a case study, an urban parking application scenario is investigated and presented. In this scenario, smart citizens collect and share parking related events, such as leaving or occupying a free parking spot. These events can be presented on a real-time city map and used in navigation software, thereby helping others alleviate parking related issues, such as the time spent while cruising for parking. The simulation environment, implemented in Java, allows to investigate and assess the critical user base of crowd-sensing based smart city applications, and the requisites (benefits, challenges) of introducing such applications. Our simulation results show that considerable gain (approx. 15%) can be achieved in the cruising time even with relatively low (30%) user base in a medium size city (Novi Sad, Serbia). Moreover, the proposed simulation environment can be used also in real field measurements by replacing/extending input data from real users.**

Keywords—Crowd-sensing; Urban Parking; Smart City; Simulation

I. INTRODUCTION

One form of volunteered co-operation between people living in smart cities of the near future is crowd-sensing [1]. In this case, the network of 'sensors' consists of people, the group of human users who are called the 'crowd'. They can collect and share a diverse body of data with their modern smart phones and devices [2]. Thus, they perform some form of sensing and thereby become citizen sensors [3]. The time is not far when people will be continuously sharing useful information in urban environments [4].

This domain, on one hand, faces challenges like handling big data, developing suitable algorithms, or maintaining security and privacy [5]. On the other hand, developing novel systems/applications which utilize crowd-sensing can be costly and the efforts can easily become unsuccessful, unless they reach the critical mass of users. Therefore it is important to perform extensive domain and risk analysis before embarking on developing a novel crowd-sensing application. One of the useful approaches in both domain and risk analysis is using simulation, i.e. creating a simplified representation of the domain of interest and developing tools for analyzing the expected behavior of the crowd in the problem domain. However, the number of existing simulation environments appropriate for inspecting crowd-sensing efforts is limited. Tanas et al. in [6] present one such simulation environment based on the ns-3 [7] network simulator. The authors inspect a scenario in which the crowd-sensing network reports randomly generated incidents in the public rail system of Barcelona. The authors also analyze the correlation between the total number of detected incidents and the number of users, i.e. the size of the 'crowd'. Crowd-sourcing, a more general concept is modeled and simulated as a complex collective intelligence on the Internet in [10].

In this paper, we introduce a simulation environment, via an urban parking case study, which can be used to analyze the possible behavior of the crowd and assess the critical user base in relation to certain crowd-sensing scenarios. We put the focus on one specific use case, namely an application, which assists users parking their cars in busy urban environments. The simulator allows the user to select a geographic region, model the occurrences of certain crowd generated events (e.g., leaving or occupying a parking spot), simulate the expected behavior of the crowd of users and investigate the gains (e.g., time spent while cruising for parking). More precisely, we focus on the investigation of the following research questions:

Q1. Can crowd-sensing users benefit from the use of the application? How? Is the benefit measurable?

Q2. Can non-users benefit from the application?

Q3. How high should the ratio of crowd-sensing users compared to non-users be in order to make the app successful?

Q4. What is the optimal ageing algorithm for crowd-sensed data, i.e. when to invalidate and set data as unusable because its age?

978-963-313-142-8 @ 2015 BME

Furthermore, the proposed simulation environment can be used instead or beside real field measurements, by replacing/extending data collected by real users.

The rest of the paper is structured as follows. In Section II, we overview some related approaches. The investigated urban parking scenario is introduced and described in Section III. We present the proposed simulation environment and its implementation in Section IV. Section V and VI contain our simulation results and their discussion, respectively. Finally, we conclude the work and present our future plans in Section VII.

II. Related work

Simulating crowd-sensing scenarios is useful to investigate the possible future behavior of the crowd and estimate whether a suitable amount of information might be collected to provide a pre-specified level of service quality, e.g. ascertain that a future crowd-sensing application will solve a problem for the end user.

In [6], the authors present a simulation environment based on the ns-3 [7] network simulation toolkit. They try to predict how the crowd would sense events in the railway system of Barcelona, and assess how many users would be necessary in order to report the majority of significant events, which are otherwise not monitored. In [8], a simulation environment is described in which the crowd collects information about parking spots. The underlying engine is custom built.

The subject of sensing in both scenarios is discrete events, i.e. the absence or presence of a particular event at a particular location and at a particular time. However, the principal difference between the above two pioneering crowd-sensing simulation efforts lies in the following fact. In [6], the crowd might form small clusters of users who are in the vicinity of an event which is sensed, while in [8] the crowd is formed from independent "agents" only circumstantially affected by the actions of other agents in their proximity (e.g., drivers will not go to a parking lot reported to be full). Unfortunately, both tools have only limited capabilities which make hard to investigate more complex scenarios.

III. Simulation scenario

The goal of the simulation environment proposed in this paper analyzes the behavior of individual drivers traveling by car in urban areas, looking for parking near their destinations. When an individual driver changes position he frees up a parking spot, and in turn when he arrives at the destination occupies another one. The simulation is influenced by individuals participating in data collection via crowd-sensing because they report all of their parking events, which can guide other individuals to find free parking spots faster.

The simulation starts after the domain model is built. The simulator creates the number of individuals given by one of the input parameters. They commute by car between the places of the five place types specified in the domain model, such as home, work, store, pub and church. The individuals leave a place and approach another one randomly, based on a predefined statistical model.

A. Movement model

The probabilities governing the choice of the next destination were configured as shown in TABLE I. The simulation differentiates two day types, namely workdays and holidays (however, holiday simulations haven't been implemented, yet). The most probable length of time spent at one specific location (e.g. work or home) is shown in the third column. The simulation was configured so that each individual goes home after the daily activities (work or work followed by shopping, or work followed by going out), latest at 24:00 and there are no further activities until 06:00 the following day.

TABLE I. Next destination selection probabilities

Activity	*Interval [h]*	*Average Length [h]*	*Workday*
Go to work	06:00 – 10:00	8	90%
Go home	14:00 – 18:00	8	80%
Go shopping	10:00 – 14:00 14:00 – 18:00	2	10%
Go out	14:00 – 24:00	2	10%
Religious institution	09:00 – 13:00	1	0%

For each individual the complete daily schedule is generated in advance based on the simple probabilities chosen by the authors and shown in TABLE I. The skeleton of the schedule generation algorithm for working individuals is as follows:

- On workdays, select 90% of individuals and choose a start of their working day from the predefined time interval. The remaining 10% of drivers is considered to stay at home.
- Generate a random length of the workday for those individuals who go to work between 07.00 and 09.00 (8+/-1).
- Send 80% of working individuals home after work.
- Send 10% of working individuals shopping after work.
- Send 10% of working individuals to go out after work. 'Pub crawls' are not supported, i.e. only one entertainment facility can be visited by an individual within the same day.

When the drivers switch locations, they look for the nearest unoccupied parking spot. If they do not find available parking in the nearest vicinity of their destination, they increase the radius of their search and 'cruise for parking' until they manage to park their vehicles. During their drives, and more precisely at the start and/or end of their drives, the drivers might share the following types of crowd-sensed events:

- Parking spot occupied;
- Parking spot left;
- Parking full – this event is created when a driver selects a parking, drives to it and learns that it is full;
- Cruising – while looking for parking;
- Parking failed – all parking lots fulfilling the selection criteria (e.g. within 500 meters) full.

For simplicity, the drivers, who share the *parking failed* event, will remain at the selected location, but their cars will not be assigned to parking lots. Optionally, in a real-life scenario drivers might share additional information, like letting others know that there are additional free spots at a certain parking lot.

The added benefits which crowd-sensors might enjoy in this simulation scenario are the following:

- Steer to parking – query the system whether there are empty spots at their destination of choice, e.g. while travelling to their next destination, they could iterate through the nearby parking lots and select the one which most probably has empty spaces.
- Steer clear of congestion – immediately cancel looking for a parking at locations where there are no suitable parking spots. This is made possible if the system 'knows' that there are no empty spots near to the chosen destination. The driver is spared of pointless cruising for parking.

The 'steer clear of congestion' perk can lower cruising time between parking lots which are known to be entirely full, thereby lowering traffic congestion and air pollution. The latter is applicable only if the driver is driving a vehicle with an internal combustion engine.

IV. Simulation Environment

Our simulator is implemented in Java and uses the services of OpenStreetMap (OSM) [10] to display simulation outputs, and MASON [11] as the underlying simulation toolkit. MASON, a multi-agent simulation toolkit, was chosen because of the wealth of available documentation, ease of development, high performance and free availability of source code. Fig. 1 presents the high level components making up the simulation environment. It utilizes shape files from Mapzen [12] and

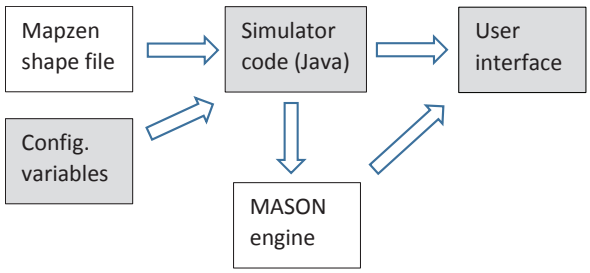

Fig. 1 Components of the simulation environment

MASON as third party components. We have designed and implemented the shaded components. The User interface integrates OSM maps and data. The flow of information between the elements is presented by the arrows.

The operation of the simulator is divided into three phases: configuration and building the domain model (e.g., urban parking); running the simulation; and generating output.

A. Configuration

As the input parameters of the simulator: i) the geographic region; ii) the number of individuals; iii) the number of individuals taking part in data collection via crowd-sensing; and iv) the length of the simulation have to be specified. To build the domain model the simulator loads the significant objects (e.g., parking spot, housing block, church, pub) of the selected geographic area from OSM shape files downloaded from the Mapzen weekly OSM metro extracts service. Mapzen was chosen over the Overpass interface [12] for querying OSM data because it creates ESRI shape files supported by MASON, more specifically by GeoMASON [14].

B. Simulation

The simulator was implemented in the Java programming language and it consists of the following three main classes (as dictated by the philosophy of the MASON simulation toolkit): Driver, Drivers and DriversWithGUI.

1) Driver

The Driver class models one individual driver who is traveling in the selected urban environment. Its behavior is governed by the statistical model described in Section III.A. These individuals belong to two distinctly different groups: crowd-sensors sharing information about parking related events and others, who do not share events.

The Driver class holds information about the geographic locations assigned to it. Each Driver instance has a statically assigned home and work location, while the rest of the locations are selected randomly from the set of available locations. The class itself implements the next destination selection algorithm described in Section III.A (details available in TABLE I.).

Note, that in this work we were not interested in monitoring the exact path and travel times of the individuals between the locations. Instead it was supposed that it takes only one simulation step (e.g. one minute) to arrive at the next destination, regardless of the distance. Similarly, when a driver arrives at an occupied parking spot, he or she will travel to the next nearest parking spot and the travel will take exactly one step (i.e. one minute).

2) Drivers

The Drivers class contains the simulation state, loads the building information from the OSM shape file, and creates and maintains the individual driver instances, i.e. agents in MASON parlance. The agents are stepped, i.e. they are triggered and might change location every minute. They might

drive to their next destination in accordance with the statistical model presented in Section III.A.

This class maintains the geographical locations associated to each individual Driver instance. It also contains the data sensed by the crowd about the occupancy of the parking spots and lots in the urban area analyzed. The drivers using the crowd-sensing application do not suspect the parking spot occupancy information received from the system, i.e. they regard them as 100% precise. This behavior will sometimes cause failed parking events.

The 'age' of each sensed information is also maintained, i.e. the exact age in minutes is memorized for each crowd-sensed event of occupying or leaving a parking spot. This is relevant, because parking spots in busy urban environments often change hands within seconds – one of the weak points of OpenSpot [15], a similar application operated by Google, was its limited ageing algorithm. Therefore events older than two minutes are tagged as unreliable, and events older than five minutes are removed from the system. These numbers are configurable, but at the moment of writing cannot be set to be less than one minute, which is the length of a single step in the simulation environment.

Driver instances which tried to park in an occupied parking spot, continue to cruise for parking towards the nearest parking spots and will spend one minute on their way. The number of parking attempts is also memorized as a significant measure of the system.

3) DriversWithGUI

The DriversWithGUI class implements the graphical interface component of the simulation environment. It presents the user a simple view of the urban area analyzed by drawing the buildings as polylines and marking the current location of drivers with a specific color (red in Fig. 2).

C. Output

The output of the simulation is: i) the sequence of the occurrences of the parking events, such as leaving a parking spot, occupying a parking spot, unsuccessful parking; and ii) the occupancy changes of the parking spots as time passes. These are saved in XML (eXtensible Markup Language) format and can be displayed on a city map.

V. Simulation Results

This section presents the simulation results measured during the experiments conducted with the simulator described in the previous section.

A. Simulated real urban areas

The simulation scenario presented in Section III was tested on freely available, real-life geographic data of two urban areas downloaded from the Mapzen service, which extracts OSM data into static shape files. The shape files of the following urban areas were used during the tests:

- Nis (Serbia)
- Novi Sad (Serbia)

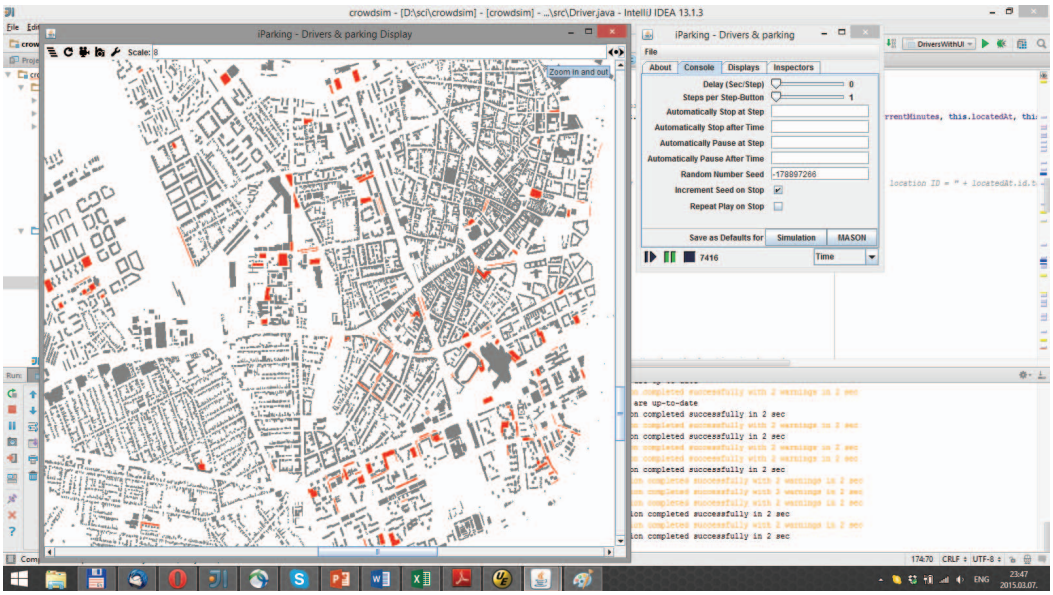

Fig. 2 Simulator interface – map based representation on the left (parking lots with crowd-sensed information colored in red), MASON console on the right (the simulation is paused)

Nis was chosen as the only city in Serbia whose OSM data was available in Mapzen at the start of the experiments. Nis was also an excellent choice as its OSM coverage is limited, with a relatively low number of both buildings and parking lots, making algorithm debug runs and tuning a little bit easier. Novi Sad was chosen as a medium sized urban environment with good OSM data coverage and quality. Its shape file was included in Mapzen output on the authors' request.

Buildings and parking lots were extracted from the shape files. The input file statistics are available in TABLE II. Where there were no available parking capacity information, defaults were set, e.g. twenty parking spots were assigned to each parking area. For simplicity, street parking locations, which are usually available in the streets in urban areas, were not taken into consideration.

Only those locations were selected, which had parking lots nearby – where 'nearby' was configured to a different distance for the two urban areas (e.g. 1 km in Nis, 500 m in Novi Sad), depending on parking coverage. The number of drivers was selected to be roughly similar to the total number of available parking spots. This ensured that there would be failed parking events and that their number could be evaluated.

TABLE II. URBAN AREA STATISTICS IN SIMULATION

Feature count	*Nis*	*Novi Sad*
Total inhabitants[1]	183,164	250,439
Residential	80	1873
Work	44	978
Shopping	5	0
Entertainment	21	115
Worship	1	10
Parking lots	20	387
Parking spots	400	7740
Total drivers	400	5000
Total journeys	~18,000	~290,000
Total cruise time [min]	~23,000	~840,000

The last three rows of TABLE II. list the pre-configured total number of drivers and the total journeys undertaken and total cruise time spent on average.

B. Simulation settings

Each simulation run during the tests lasted 30 days, i.e. one month. The time resolution, i.e. the period between the steps in MASON terminology was one minute. The ratio of drivers sharing parking related events (i.e. crowd-sensors) to those not participating in these activities was varied during the tests between 0% and 50% (see TABLE III. for details).

[1] Statistical data freely available on Wikipedia (March, 2015)

Each test was run three times and the results were averaged. MASON generated consistent results, with no striking outliers in neither of the simulation runs. Simulation runs took about one minute for the city of Nis with MASON's simulation step configured to 1.0 (i.e. default). The simulation times were comparable to the number of agents, i.e. they were considerably longer for the city of Novi Sad.

The crowd-sensing based system calculates perceived parking availability with two possible decision support values, (probably) available and unavailable, in the following manner:

- It was supposed that parking lots are not monitored, i.e. not aware of available capacity in real-time;
- Lots were optimistically regarded as free by default;
- Each *parking full* event within the last sixty minutes marked the lot as probably full;
- Each *parking spot occupied* event collected within the last sixty minutes marked the lot as probably available – if not invalidated by a parking full event;
- Crowd-sensors chose the first nearby parking spot proposed by the system, or a random spot if the system did not have suggestions, e.g. when the system 'thinks' that all suitable lots are full.

All crowd-sensed information was treated as valid, i.e. the potential appearance of malevolent crowd-sensors sharing false or misleading information was not taken into consideration.

C. Collected measurements

The most important measurements collected during the experiments were the following:

- Cruising time – the time elapsed while travelling from a full parking lot to the next chosen parking lot. This is in line with the number of parking trials;
- Failed parking – no free parking found in the vicinity of the chosen destination;
- Total journey ends – the total number of trips ended by the drivers;
- Total events collected by crowd-sensors.

Each journey undertaken by the drivers could end in two ways: either the driver found an empty parking lot near his destination, or he failed to park.

The rule for calculating cruising time was the following: when a driver 'hits' a full parking lot, he chooses the next suitable parking lot and elapses one minute cruising, regardless of the distance. The number of total events collected is an important measure in forecasting the expected data load the system would face in a real life environment.

Fig. 3 presents the simulated effect of crowd-sensed information on the measured total *cruise time* events in the Novi Sad scenario as perceived by both the crowd-sensors and

non-sensors. They were normalized by the total number of journeys in order to show them in the same diagram.

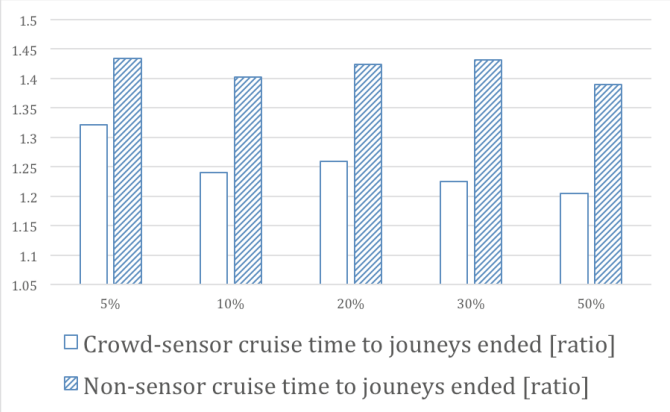

Fig. 3 Cruise time compared to journeys ended, Novi Sad scenario

The ratio of cruise time to the total number of journeys was lowered by up to 14% when comparing crowd-sensors to non-sensors. This improvement can be seen by comparing the height of the columns in Fig. 3. The exact comparison results for both scenarios are shown in TABLE III.

TABLE III. MEASURED CRUISE TIME OPTIMIZATION FOR CROWD-SENSORS COMPARED TO NON-SENSORS

Ratio of crowd-sensors	*Nis*	*Novi Sad*
5%	3.1%	7.8%
10%	3.2%	11.6%
20%	3.3%	11.6%
30%	7.5%	14.4%
50%	4.8%	13.3%

The amount of *failed parking* events was selected as a negative performance indicator. Failed parking events occur when a driver chooses a parking place, drives to it and finds it entirely full. These events remained roughly the same during the simulations in the Novi Sad scenario, ranging between 0.41 and 0.43, for both crowd-sensors and non-sensors (see Fig. 4). The values shown in Fig. 4 were also normalized with the total number of journeys ended.

The fact that parking failed events occurred in the simulations might seem strange, considering that the total number of drivers and parking spots is equal in the Nis scenario, and lower in the Novi Sad scenario (i.e. there are more parking spots than drivers). The explanation of this phenomenon is that the drivers flock to busy parts of the cities where there are insufficient numbers of parking spots, while in other, less frequent parts of the cities there are empty spots.

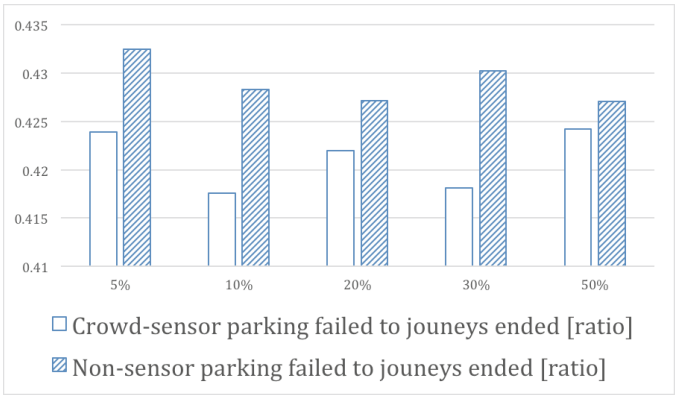

Fig. 4 Parking failed count and journeys ended ratio, Novi Sad scenario

The lowered cruise time (3-14%) measured during the simulation scenarios might seem as a relatively low value, but considering that in the very small Nis simulation scenario the 400 drivers cruised for 19,000 minutes per month on average, this small improvement would allow the city to save hundreds of driver hours per month on the city's streets.

The relatively high number of failed parking events registered during the tests with the relatively modest OSM data available for the city of Nis, might be attributed to the low number of parking lots and their relatively weak distribution, i.e. the destinations were clustered around a few, very busy parking lots in the city center, where most of the drivers travelled and tried to park their cars.

The last value measured, namely the amount of events sensed and shared by the crowd was relatively large even for the smaller crowd-sensor to non-sensor ratios as crowd-sensors shared almost 200 parking related events per month in the Nis scenario. This obviously means that for small user bases the total number of monthly events is measured in a few thousands, but for a few thousand active users the number of sensed pieces of information would reach millions, providing a wealth of information, which might be a basis for developing additional applications and optimization.

VI. DISCUSSION

This section contains a retrospective on the four research questions raised in the Introduction and discusses the experimental results presented in Section V.

A. Crowd-sensor benefits

Through the first research question we analyzed whether crowd-sensors would benefit from using the proposed crowd-sensing based application, and their benefits would be measurable. The simulation results showed that crowd-sensors would have a measurable benefit from using the proposed crowd-sensing based urban parking optimization application. The cruise time, i.e. the time spent on the road after a failed parking event was lowered in all but one simulated scenario, by 3% to 14%. At the same time the crowd-sensors did not face

adverse effects, as measured through the superficially changed count of failed parking events.

B. Non-sensor benefits

The second research question asked whether non-sensors, i.e. the drivers, who are not participating in crowd-sensing and not using the decision support application, would still benefit from the application. The simulation results showed that the amount of cruise time spent by non-sensors was not considerably affected, i.e. they would not benefit or suffer from the crowd-sensed app. On the other hand they would not face adverse effects either, as measured through the unchanged number of parking failed events.

C. Necessary user base

Via the third research question we analyzed the crowd-sensor to non-sensor ratio necessary for the crowd-sensing application to become successful. In the urban parking scenario, this was formalized as follows: how high should the crowd-sensor driver percentage (as compared to the number of all drivers) be in order to collect sufficient amounts of information necessary to steer application users away from full parking lots?

Time spent cruising between two parking lots was taken as a measure of success. The lower the time spent cruising, the better for the operation for the whole urban transport system, as well as for the natural environment, which is less affected by exhaust fumes.

The total cruise times for both crowd-sensors and non-sensors were normalized by dividing the total number of journeys. The simulations showed that the crowd-sensors start feeling the benefits with as low as 5% coverage, when the benefit was 3.1% to 7.8% less time spent with cruising. The highest benefit of 14.4% were measured for 30% coverage (see row '30%', column 'Novi Sad' in TABLE III.).

D. Crowd-sensed measurement ageing

The fourth research question deals with the analysis of the best ageing algorithm for crowd-sensed data, i.e. when to invalidate sensed data and flag it to be too old and not useful anymore. During the experiments the ageing time was varied between 10 and 60 minutes, but no firm conclusions could be made based on the measurements. Based on the initial experimental results, it can be assumed, that the aging algorithm should be dynamic and tailored, taking into consideration both the size of the urban area, the number of total drivers expected and the time of day.

The choice of proper ageing algorithms in combination with a forecasting algorithm based on historical crowd-sensed data might be the way to go for crowd-sensing applications.

VII. Conclusions

This paper presents a simulation environment developed for analyzing crowd-sensing based applications in the Smart City application domain. An urban parking application scenario was investigated and presented as a case study. In this scenario, smart citizens collect and share parking related events, such as leaving or occupying a free parking spot.

The simulation environment itself was developed in the Java programming language using the MASON multi-agent simulation toolkit. The simulator allowed the authors to investigate and assess the critical user base of a crowd-sensing based smart city application, more specifically an application assisting drivers finding parking spots in urban environments.

The simulation runs executed during this work showed that the users of the crowd-sensing application start feeling the benefits with modest sized user bases of 5% crowd-sensor to non-sensor ratios. The benefit was measured as the decline in total cruise time spent while looking for parking. The best results were achieved for the Novi Sad simulation scenario with 5000 agents and 30% crowd-sensor participation, i.e. when 1500 drivers were sensing and sharing parking related events and enjoying the benefits of ~14% shorter cruising times.

As a continuation of this research, the authors intend to optimize the simulator, which will enable them to experiment with higher number of agents in large urban environments (e.g., in Budapest size cities). The authors plan to extend the urban parking scenario with a more detailed assessment of cruise times, i.e. more precisely calculated cruise times spent while cruising for parking, instead of the fixed one minute cruise times used in this paper. The authors also plan to generalize the simulator so that it might be used for simulating crowd-sensing enabled solutions in other domains.

Acknowledgment

This research was supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia under grants III-42004 and TR33013, and by the Hungarian Academy of Sciences under grant Domus 105/6864/HTMT. Károly Farkas has been partially supported by the Hungarian Academy of Sciences through the Bolyai János Research Fellowship. This work was supported by the COST Action IC1203 (ENERGIC).

References

[1] D.C. Brabham, "Crowdsourcing the public participation process for planning projects," Planning Theory, vol. 8(3), pp. 242–262, 2009.

[2] R.K. Ganti, F. Ye and H. Lei, "Mobile crowdsensing: current state and future challenges," IEEE Communications Magazine, vol. 49(11), pp. 32–39, 2011.

[3] M.F. Goodchild, "Citizens as sensors: the world of volunteered geography," GeoJournal, vol. 69(4), pp. 211–221, 2007.

[4] F. Zambonelli, "Pervasive urban crowdsourcing: visions and challenge," 2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), Seattle, USA, pp. 578–583, 2011.

[5] L. Díaz et al, "Managing user-generated information in geospatial cyberinfrastructures," Future Generation Computer Systems, vol. 27, pp. 304–314.

[6] C. Tanas, and J. Herrera-Joancomart, "Crowdsensing simulation using ns-3," Citizen in Sensor Networks, Lecture Notes in Computer Science, pp 47-58, 2014.

[7] ns-3 Simulator, http://www.nsnam.org, accessed Mar 2nd, 2015.

[8] D. Centola, "The Ascape model developer's manual," http://ascape.sourceforge.net/manual/Overview.html, accessed Mar 2nd, 2015.

[9] T. Buecheler, R. Lonigro1, R.M. Füchslin, and R. Pfeifer, "Modeling and simulating crowdsourcing as a complex biological system: Human crowds manifesting collective intelligence on the Internet," The 11th European Conference on the Synthesis and Simulation of Living Systems, vol. 11, MIT Press Boston, 2011.

[10] M. M. Haklay and P. Weber, "OpenStreetMap: user-generated street maps," IEEE Pervasive Computing, vol. 7(4), pp. 12–18.

[11] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan and G. Balan, "MASON: A multi-agent simulation environment," Simulation: Transactions of the Society for Modeling and Simulation International, vol 82 (7), pp. 517-527, 2005.

[12] Mapzen, "Weekly OSM metro extracts," https://mapzen.com/metro-extracts/, accessed Mar 2nd, 2015.

[13] OpenStreetMap Wiki, "Overpass API", http://wiki.openstreetmap.org/wiki/Overpass_API, accessed Mar 2nd, 2015.

[14] Sullivan K., Coletti M., Luke S., "GeoMason: Geospatial support for MASON," George Mason University Technical Report GMU-CS-TR-2010-16, 2010.

[15] Sherwin I., "Google Labs' open spot: A useful application that no one uses," http://www.androidauthority.com/google-labs-open-spot-a-useful-application-that-no-one-uses-15186, accessed Mar 2nd, 2015.