Data Aggregation and Roadside Unit Placement for a VANET Traffic Information System

Christian Lochert[⋆] Björn Scheuermann[⋆] Christian Wewetzer[⋄]
Andreas Luebke[⋄] Martin Mauve[⋆]

[⋆]Computer Networks Research Group
Heinrich Heine University
Düsseldorf, Germany

{lochert,scheuermann,mauve}@cs.uni-duesseldorf.de

[⋄]Volkswagen Group
Wolfsburg, Germany

{christian.wewetzer,andreas.luebke}@volkswagen.de

ABSTRACT

In this paper we investigate how a VANET-based traffic information system can overcome the two key problems of strictly limited bandwidth and minimal initial deployment. First, we present a domain specific aggregation scheme in order to minimize the required overall bandwidth. Then we propose a genetic algorithm which is able to identify good positions for static roadside units in order to cope with the highly partitioned nature of a VANET in an early deployment stage. A tailored toolchain allows to optimize the placement with respect to an application-centric objective function, based on travel time savings. By means of simulation we assess the performance of the resulting traffic information system and the optimization strategy.

Categories and Subject Descriptors

H.4.3 [**Information Systems Applications**]: Communications Applications; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*distributed applications*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*distributed networks*

General Terms

Algorithms, Design, Performance

Keywords

VANET, Aggregation, Genetic Algorithms, Information Dissemination

1. INTRODUCTION

Dissemination applications based on VANETs face two key challenges: a limited network capacity shared by all cars and—at least initially—a highly partitioned network limiting the speed of data dissemination. In this paper we tackle these problems in the context of a cooperative traffic information system, where all participating cars gather data on the current traffic situation. This information is then distributed so that other cars may use it for improved route planning.

It is obviously not possible to distribute all data to all cars. This would quickly exceed the available bandwidth. Instead, we propose to aggregate it in a hierarchical fashion: the farther away a region is, the coarser will be the information on its traffic situation. The general idea of hierarchical aggregation is not new. However, existing hierarchical aggregation schemes focus on combining data from geographical regions, with aggregates representing averages or extremal values within these regions. This may be fine for aggregating information like the availability of parking places. It is useful to have an aggregate describing the total number of available parking places within some region. For a traffic information system, however, it is not sufficient to know geographical averages or extremal values. We therefore propose a hierarchical aggregation scheme for travel times in road networks. Essentially, we use coarser and coarser approximations of the road network to summarize travel times in regions that are farther and farther away.

The aggregation scheme allows to deal with network capacity limits by summarizing the collected data. But VANETs also suffer from very limited connectivity and from limited information propagation speed, especially during early rollout. It has been discussed in prior work to improve information dissemination by adding comparatively inexpensive infrastructure at some locations in a city. These supporting units (SUs) exchange information with cars passing by and, using a backbone network, with each other, thereby delivering information rapidly to more distant regions. However, while it has been shown that even a very limited number of SUs can largely improve the dissemination process, it is not yet clear where they should be located. Therefore, as the second major contribution in this paper, we present an optimization methodology to find a good placement for SUs in the context of a VANET-based traffic information system. The presented approach makes use of a genetic algorithm to maximize the travel time savings of cars in a city environment.

The remainder of this paper is structured as follows. In Section 2 we review related work. Following that, we introduce in Section 3 our scheme for aggregating travel time information in city environments. In Section 4 we show how a combination of good SU locations for a traffic information system on the basis of this aggregation scheme can be found. We present and discuss the results of a simulative evaluation in a VANET city scenario in Section 5. Finally, we conclude this paper with a summary in Section 6.

2. RELATED WORK

Many of the recently proposed VANET travel comfort or traffic flow applications rely on some form of data dissemination and aggregation. In this section we discuss four applications that stand exemplary for a specific idea and show how they deal with the problems related to the work presented here.

In the Self-Organizing Traffic Information System (SOTIS) [12], information is distributed by sending periodic beacons containing information on the traffic situation in a larger surrounding. The authors outline a non-hierarchical aggregation scheme. It combines information from different sources on the same road segment into one average value. This helps to reduce the transmitted data and is well-suited for one-dimensional highway scenarios, but it is not sufficient in complex city scenarios, where many possible routes exist between two points and the network capacity does not allow for a distribution of separate data items for each single road segment in the whole city.

TrafficView [8] is another system for disseminating traffic information, similar to SOTIS. It also introduces a data aggregation scheme. TrafficView distributes information on position and speed of single vehicles. The aggregation mechanism combines a number of “similar” vehicles in an adaptive way, aiming to minimize the introduced errors. Like SOTIS’ aggregation approach, this does not overcome the problem of insufficient capacity to provide individual information on each road segment.

In [2], a VANET application distributing information on free parking places is presented. Here, too, the information is disseminated by periodic broadcasting. Aggregation is performed hierarchically in a quad-tree grid structure over the city area. Such an area-based aggregation scheme cannot be used to summarize travel time data in a road network.

A probabilistic algorithm for the hierarchical aggregation of observations in VANET-based traffic information systems is presented in [7]. The aggregates are based on sketches, i. e., on probabilistic approximations instead of exact values. This allows to update and merge aggregates in a duplicate-insensitive way, thereby improving the aggregate quality. How an aggregate can be represented is the central contribution in that work. This is complementary to the work presented here: in our aggregation scheme, we focus on what data should be combined into an aggregate, instead of the data structure carrying the actual value.

The general feasibility of information dissemination in city scenarios is analyzed and discussed in [6]. There we propose to use specialized infrastructure to improve dissemination performance, called “stationary supporting units”. We demonstrate that a small number of networked supporting units improves the dissemination performance dramatically. However, realistic VANET applications have not been considered in the evaluation, and the placement of SUs followed simple heuristics. Here, we take up the idea of SUs and infrastructure supported dissemination techniques, and present an approach for optimal SU placement in the context of a traffic information system.

A substantial amount of work has been put into optimizing the beaconing in dissemination applications. This includes adaptive beaconing intervals, and the selection of information to be incorporated into a beacon of limited size. Work in this area can, for example, be found in [1, 2, 8, 12]. VANETs also face important security questions. It has, for instance, been studied how VANETs and aggregated messages in them can be made tamper-proof or resistant to cheaters [4, 5, 9, 10]. Solving these issues is largely orthogonal to answering the question how and which aggregates should be formed, so these ideas can often be combined with our approach.

3. AGGREGATION

The basic idea of our aggregation scheme is as follows: we define landmarks on multiple levels of a hierarchy in the road network. At the highest level these are junctions of the main roads or highways. Lower levels include all higher level landmarks plus more and more intersections of smaller streets. The lowest level is a representation of the full road network. Cars passing a road segment can make an observation of the current travel time between two neighboring landmarks. This information is distributed within the closer surrounding. It is used by cars to calculate travel times between landmarks of the next higher level, thereby summarizing the travel times in the area. This coarser picture on the travel times is distributed in a larger area than the observations of individual cars. It is also used to calculate the travel times between landmarks of the next higher level of the hierarchy, and so on.

These aggregation steps are performed by the cars themselves, in a completely decentralized fashion, whenever information that is a suitable basis for forming an aggregate becomes locally available.

3.1 Aggregation on a single level

Figure 1 shows an example for a single hierarchy level. The travel time between the landmarks *Eiffel Tower* and *Arc de Triomphe* is determined. These two high-level landmarks are connected via a number of possible routes over landmarks on the next lower level, here indicated by circles. The travel times between these lower-level landmarks are known, either from direct observations or from received or previously calculated lower-level aggregates. The aggregated travel time from Eiffel Tower to Arc de Triomphe is the travel time along the minimal travel time route between these two points. Essentially, this compresses all information on all possible paths between two landmarks to a “virtual” link between both landmarks.

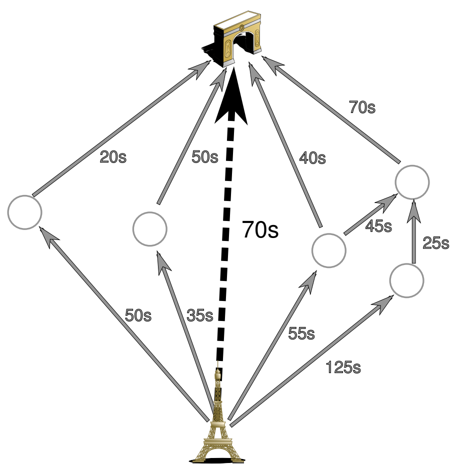


Figure 1: Landmark aggregation.

This approach can be stated more formally as follows. The road network can be seen as a directed graph $G(E,V)$ consisting of junctions $v \in V$ and street segments $e \in E \subseteq V^2$ connecting these junctions. The segments are rated with a weight $w(v_1,v_2)$ corresponding to the current travel time. Some junctions are distinguished as landmarks $l \in L, L \subseteq V$. A route $r(A,B)$ between two landmarks A and B is a sequence of junctions $(v_1,\ldots,v_n)$ such that $v_1 = A$, $v_n = B$ and all pairs of consecutive junctions are connected by a street segment, i. e., for all $i = 1,\ldots,n-1$ there exists $(v_i,v_{i+1}) \in E$. The cost of this route is

$$\|(v_1,\ldots,v_n)\| := \sum_{i=1}^{n-1} w(v_i,v_{i+1}).$$

Let $R(A,B)$ denote the set of all possible routes from A to B. We can then define the fastest route $r^*(A,B)$ as follows:

$$r^*(A,B) := \operatorname*{argmin}_{r \in R(A,B)} \|r\|.$$

$\|r^*(A,B)\|$ is used as the travel time between landmarks A and B on the next higher level. I. e., the operation performed when calculating an aggregate is to determine $\|r^*(A,B)\|$ based on lower-hierarchy travel times. Any standard routing algorithm may be used in order to calculate $\|r^*(A,B)\|$. Note that it is *not* relevant which route actually achieves this travel time. While the car is further away from both landmarks, the relevant information is that it is *possible* to travel from A to B within the given time. When it comes closer to the respective area, it will receive the locally available, more detailed information. This information can then be used for routing.

Aggregated travel times should not be calculated for each pair of landmarks. First, this would result in a number of aggregates that grows like $O(n^2)$ with the number of landmarks. More importantly, however, travel times aggregates between landmarks that are very far away from each other do also not contribute much additional information: there will be many other landmarks "in between", and a sequence of aggregates over those is likely to be a good approximation of the travel time between the distant pair of landmarks. Hence, only such pairs should be considered that are not too far apart. It is either possible to use a fixed criterion, like a maximum beeline distance of landmark pairs for which aggregates are formed on a given hierarchy level, or to mark the landmark pairs explicitly in the map data, choosing them such that a good approximation of the underlying road network is maintained. In our evaluation here, we follow the latter approach.

3.2 Judging the quality of information

Before calculating the aggregated travel time between two landmarks and passing it on to other cars, a car needs to be able to judge whether its knowledge about the current traffic situation suffices for a good estimate. From a very general perspective, a very large number of road segments could lie on a possible route between two landmarks. It is therefore necessary to determine which road segments are likely to be relevant for the travel time estimate. An aggregate may be formed if information on these relevant road segments is locally available.

In order to define the set of relevant road segments, we look at what we call the *standard travel times* along the road segments. These travel times are hard-coded in the road map data and represent reasonable expectations of the travel times, as they are currently used for non-dynamic road navigation systems. One can easily calculate the *optimal standard route* $r^*_{\text{std}}(A,B)$ from A to B on the basis of this static data—this is essentially what current navigation systems do. For any route r, it is also easily possible to calculate the standard travel time $\|r\|_{\text{std}}$.

We then choose a threshold $\theta > 1$. We define the set $\mathscr{R}(A,B)$ of relevant road segments between two landmarks A and B to encompass all road segments that lie on a route for which the standard travel time is at most by a factor of θ longer than the optimal standard travel time. I. e., a road segment e is in $\mathscr{R}(A,B)$ if and only if there exists a route $(v_1,\ldots,v_n)$ from A to B such that e is part of that route and

$$\|(v_1,\ldots,v_n)\|_{\text{std}} \leq \theta \cdot \|r^*_{\text{std}}(A,B)\|_{\text{std}}.$$

Since this criterion is based only on the (static) standard travel times, the set of relevant road segments does not depend on the current traffic situation or on a car's current knowledge. We allow the calculation of an aggregated travel time from A to B if information on the road segments in $\mathscr{R}(A,B)$ is available.

3.3 Hierarchical aggregation

In order to perform hierarchical aggregation, landmarks are assigned a level in a hierarchy. Landmarks of a higher level are also members of all lower levels. More formally, for a set of landmarks L_i of an aggregation level i:

$$L_i \subset L_{i-1} \subseteq V, \quad i > 1.$$

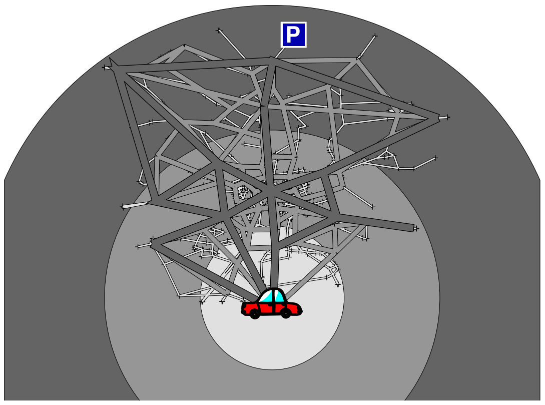
To form an aggregate on hierarchy level i, the aggregates of level $i-1$ are used in the very same way as individual observations of cars are used on the first level. Thus, the landmarks on level $i-1$ are used like the junctions in the discussion above, and the aggregated travel times between them take on the role of the travel times along individual street segments.

The area in which individual observations and aggregates are distributed is limited based on their level in the hierarchy. Individual observations are distributed in a very limited range whereas the highest level aggregates are distributed in the whole network. Figure 2(a) depicts this. The driver of the car located in the south of the scenario intends to travel towards the north. Three hierarchy levels can be seen here. The circles around the car indicate the regions from which detailed, fine-grained level 0 information, slightly aggregated level 1 information, and more coarsely aggregated level 2 information is available to this car. As the car travels towards its destination, these regions shift as shown in Figure 2(b), as additional information is received.

The destination of a trip will not always be a high-level landmark position. Nevertheless, the aggregated information can of course be used for route planning. In order to do so, a navigation system would "fill up" the missing information between the final destination and close-by landmarks by using the standard travel times hardcoded in the map data. This is reasonable, because a final decision on the last part of the route is not yet required at this stage—it is sufficient if a good choice for the immediately upcoming routing decisions can be made. While the car approaches its destination, the route can be updated and refined as more detailed information becomes available.

4. PLACEMENT OF SUPPORTING UNITS

During the rollout of car-to-car technology the equipment density of cars participating in a VANET will be low. This makes timely information dissemination very difficult. It is the second key problem that has to be solved for a VANET-based traffic information system. It has been proposed (e. g. in [6]) to make use of infrastructure devices—*supporting units (SU)*—to improve dissemination performance. SUs use the same radio technology and


(a) Starting point.

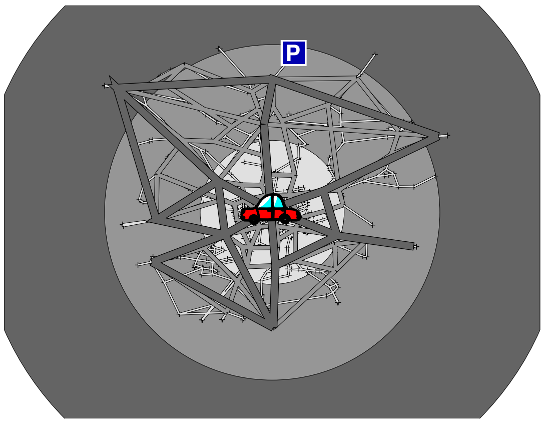

(b) During the trip.

Figure 2: Hierarchy based navigation.

essentially the same application as equipped vehicles. They are able to receive observations and aggregates from passing-by cars. They also send beacons and thereby hand over their knowledge to cars. The central benefit of SUs, however, is achieved by connecting them via a backbone network, allowing them to exchange information. This can bring up-to-date knowledge to distant network regions in very short time.

A very limited number of SUs is sufficient for substantial benefits. But nevertheless SUs incur deployment and maintenance costs. Hence, the question arises how to achieve good performance with as few SUs as possible, or—putting it the other way around—at which positions in the road network of a given city a given number of SUs should be located in order to achieve high benefit. This question is closely related to the employed aggregation scheme, because both in conjunction determine which information will be able to arrive at which location and which point in time.

In the following we propose a way to position SUs such that their application level benefit is maximized. To this end we define an application level metric for a traffic information system. This metric reflects the travel time saved by using the application. Then we show how a genetic algorithm can be used to optimize the placement with respect to this metric.

4.1 Optimizing SU placements

In a city, there are typically many possible positions for SUs. Given a set of potential SU locations and a number of SUs to be placed, our approach aims to identify the optimal subset of locations. For a given SU positioning, it is theoretically conceivable to run a simulation (using, e. g., an integrated simulation environment that models both car movement and network traffic) and to measure the achieved travel time saving. But even with a moderate number of possible locations and SUs, the number of possible combinations is overwhelming. If there are 100 potential locations for 10 SUs, there are $1.73 \cdot 10^{13}$ possible placements. With 30 SUs, there are $2.9 \cdot 10^{25}$ possibilities. Therefore it is obviously not feasible to assess and compare all placements. Identifying the optimal subset of SU locations actually turns out to be a very difficult optimization problem. Here, we use genetic algorithms in order to find a good approximation.

Basically, genetic algorithms start off with a random set of “individuals” (SU placements), assess their “fitness” (achieved travel time savings), and then generate a new “generation” of individuals by combining features of the “fittest”. This approach has been applied to a broad range of problems, and often yields excellent results. A more detailed description can be found in [11]. Nevertheless, the computational effort for assessing SU placements remains significant. Typically, at least several dozens of generations are necessary, each with many individuals. For each of these individuals the fitness—i. e., the objective function—needs to be calculated.

A fully-fledged simulation would model car movement, the network (i. e., radio propagation, medium access, exchanged beacons etc.), and the application (i. e., the contents of the beacons) in parallel. This is computationally very expensive (with standard traffic and network simulators at least a few hours of computation time per individual), and is thus still not possible for all these individuals within reasonable time. Thus, some approximations must be found, which significantly speed up the simulation, but still capture the relevant effects in sufficient detail. We propose such a method to obtain an estimate of travel time savings within a comparatively short time (typically 1–3 minutes). This is made possible by decoupling the application-layer simulation from the lower layers.

4.2 Estimating travel time savings

Dissemination takes place by periodic beaconing by both cars and SUs. Though other approaches are generally conceivable, in the majority of schemes and also in the mechanisms employed here, the transmission of these beacons does not depend on their contents or on the sender’s knowledge. PHY and MAC effects do also not depend on the data within the packets. The points in time at which the network nodes transmit and the set of nodes receiving each transmission do therefore *not* depend on the transmitted data. Therefore, we may simulate the network traffic *independently* from the application. In practice, we use VISSIM for the simulation of car movements in a city, and ns-2 for the simulation of periodic beacons issued by all cars and SUs; other traffic and/or network simulators could be used without generally affecting the proposed methodology. This simulation step yields a log file in which all beacons are recorded with their respective receiver set, along with the car positions. We may then—subsequently—run a separate *application simulator* that reads this log file, keeps track of the knowledge base of all nodes, performs aggregation as specified in the previous section, decides about the data contained in each of the beacons, and respectively updates the knowledge bases of all receivers of the beacon. This does not allow to model the impact of changing knowledge on the behavior of the cars (in particular, their route will not depend on their knowledge). But unless the large-scale traffic flow is significantly impacted by use of the application (implying that a very large fraction of cars use it), the effects will be negligible.

The so far described approach is able to separate traffic and network simulation from the simulation of the application. But the network simulation simulates SU beacons and therefore still depends on the SU positioning. This is where one of the central approximations made by our method comes into play: we simulate SUs at *all* possible SU locations in the network simulator, and let *all* of them transmit periodic beacons. In the application simulator, we then simply ignore beacons from "non-existing" SUs; the respective beacons are not considered when updating the knowledge bases. The number of SU locations is small in relation to the number of cars. So, the vast majority of network traffic is caused by cars, and occasional beacons sent at unoccupied SU locations have only negligible effects. But this simplification allows us to re-use a traffic and network simulation log file for application simulations of any arbitrary set of SU locations. A set of such log files may be precomputed. A flow-chart outlining the toolchain is depicted in Figure 3.

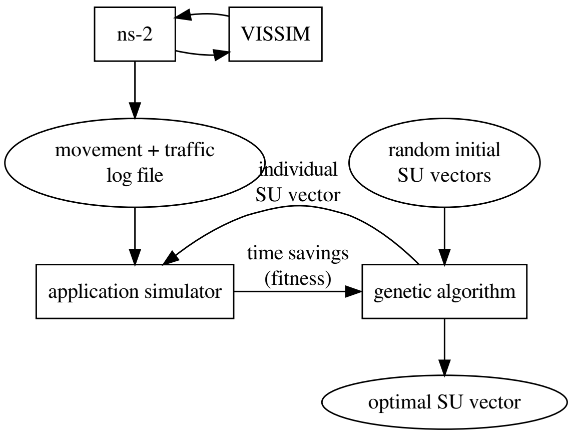


Figure 3: Toolchain used for SU placement optimization.

We can now draw samples of estimated travel time savings in the following way. The application simulator assigns travel times to all road segments. Our implementation picks a pre-configured fraction γ of the road segments and assigns them a travel time that is ϕ times higher than their respective standard travel time; the travel time of all other segments is set equal to the standard travel time. If more sophisticated models for the travel times become available, they can of course easily be plugged in.

We choose a car at a specific point in time. The car's position at this time is known, as well as its (not necessarily perfect) knowledge about the current traffic situation. We also choose a random destination for the car. We then calculate the optimal route r^* from the car's current position to this destination, based on this car's knowledge. We also calculate the optimal standard route r^*_{std} based on standard travel times as introduced in Section 3.2. This is the route a standard navigation system would choose. For both routes, we calculate the travel times based on the true current travel times as set by the application simulator. The ratio of these two travel times is used as the estimated travel time saving. Note that the car's current knowledge is typically not perfect. It will virtually always deviate from the current traffic situation to some extent (e. g., because the situation changes over time). The dynamic route might therefore even be worse than the standard route. The travel time benefit is thus highly dependent upon the dissemination performance: it will be high if up-to-date information relevant for the route calculation is known by the car. The mean travel time savings over many such samples can be used as a metric for the application benefit.

Of course, this approach is an approximation in many regards. The simulation methodology makes, as already discussed, a number of simplifying assumptions. Furthermore, for example, cars do not actually follow their dynamically chosen routes in the simulation, and thus information potentially obtained during their travel and subsequent route adjustments are not taken into account. The central benefit of the method, however, is that it allows to obtain a good estimate with limited computational effort. It is therefore usable as an objective function for the optimization of the SU placement.

4.3 Genetic algorithm

Based on the method for calculating the objective function for a given SU placement, a genetic algorithm can be used to actually find a good SU placement. To this end we need to express the set of occupied SU locations as an "individual". There is a rather natural representation for this purpose: a bit string, where each bit position stands for one SU location. A bit is set to one if and only if there is an SU at this position. An example representation is depicted in Figure 4. There are ten possible positions of SUs, four of them are used.

4.3.1 Selection and recombination

For each generation, the fitness of each individual is computed. We do so using the methodology described above. Then the sum of the fitness values of all individuals in the generation is calculated. The selection of parents for the next generation is then performed by randomly selecting individuals based on their relative fitness, i. e., their own fitness divided by the sum.

For a given, fixed number of SUs we want to optimize over all bit vectors with the respective number of bits set. Therefore, when combining two parent individuals, it needs to be ensured that the newly created individual still has the same number of bits set. Standard recombination techniques like uniform or multi-point crossover cannot guarantee this. We propose to use the following recombination technique: in the child's bit vector, we first set all bit positions that are set in *both* parents (i. e., we start with the bit-wise AND of the parents). We then "fill up" the bit vector by setting additional bits. We choose these additional bits randomly from all bits set in exactly *one* of the parents (i. e., from those bits enabled in their bit-wise XOR).

4.3.2 Mutation

After creating children, it is vital for genetic algorithms that "mutations" happen. They avoid that the algorithm gets trapped in a local optimum. This is done by randomly flipping some (few, here we use 0.4 %) of the bit positions. Again we need to make sure that the number of set bits remains constant. We therefore chose to mutate by simply exchanging the state of two bits in the bit vector.

4.3.3 Parallelization

In our implementation we use a Multi-Population Genetic Algorithm (MPGA) with two separate populations. It has been shown that this leads to faster convergence properties for genetic algorithms; for a survey see [3]. After each generation it is possible that an individual migrates to the other population.

SU_0	SU_1	SU_2	SU_3	SU_4	SU_5	SU_6	SU_7	SU_8	SU_9
1	0	1	1	0	0	1	0	0	0

Figure 4: The genetic representation of supporting units.

4.3.4 *Search termination*

We stop the genetic algorithm if we arrive at homogeneous populations, i. e., if one specific set of SU locations dominates the populations. If no homogenization occurs, we stop the algorithm after a fixed number of generations.

The operation of the genetic algorithm is summarized in Figure 5. Whenever one iteration (i. e., one generation of individuals) is finished, the individuals of the old population are replaced by newly recombined and mutated ones. In order not to lose the information of the so far overall best individual, the memorized chromosome is also included into the new population. In the last step of each iteration, the algorithm randomly chooses an individual from each population. These individuals are then moved to the other population. After this final step the genetic algorithm proceeds with the next generation.

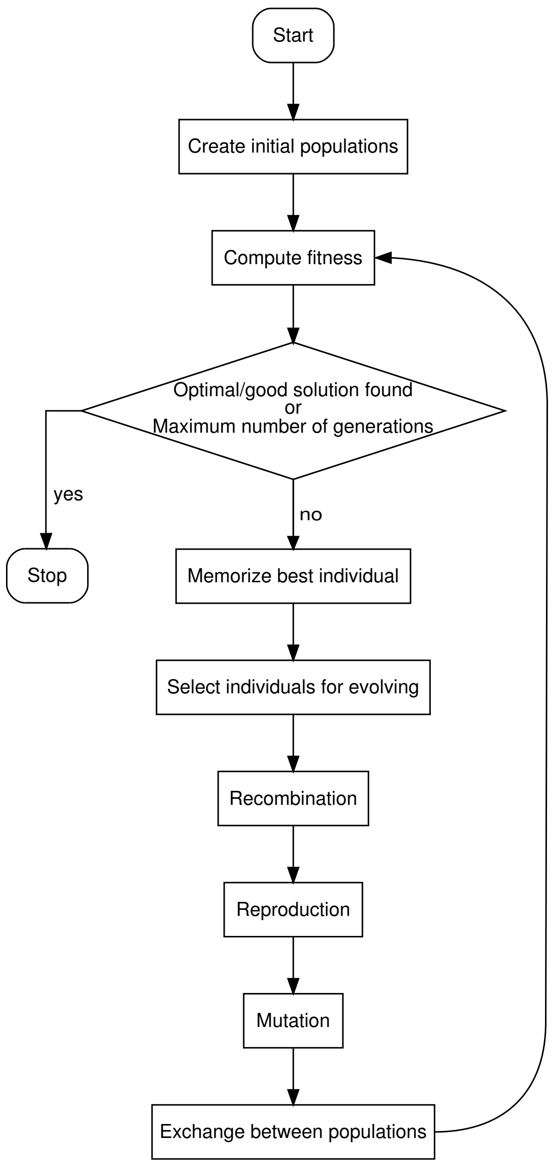

Start
Create initial populations
Compute fitness
Optimal/good solution found
or
Maximum number of generations
yes
no
Stop
Memorize best individual
Select individuals for evolving
Recombination
Reproduction
Mutation
Exchange between populations

Figure 5: The steps of a genetic algorithm.

5. EVALUATION

In order to evaluate the aggregation scheme and SU placement methodology outlined above, we apply it to a specific VANET city scenario model and analyze aspects of both the traffic information system and the SU placement approch with genetic algorithms. In a very similar way, it could be applied to find good SU placements for any other real city, given that a sufficiently detailed road and traffic model is available.

5.1 Simulation setup

We use VISSIM to simulate vehicular traffic in the city of Brunswick, Germany. The model covers an area of approximately $16 \times 16\,\text{km}^2$, 500 km of roads and a total of about 10 000 cars. It is based on detailed measurements of the real traffic pattern in the city. We use an average equipment density of 0.25 equipped vehicles per radio range. This corresponds to a VANET equipment ratio of 5 % of all cars. In ns-2, IEEE 802.11 is employed as the MAC protocol, with the two-ray ground propagation model, a communication range of 250 meters, and a carrier sense range of 550 meters. The network simulator also uses an obstacle model that does not allow radio signals to propagate through the walls of buildings.

As mentioned above, the objective of the genetic algorithm is to find the optimal vector of supporting units for 100 predefined possible locations (this could, e. g., be the positions where potential cooperation partners are located, who would allow for an SU to be installed). The genetic algorithm starts with 40 individuals, split into two populations. Each simulation run of the application simulator uses a randomly chosen random seed. This ensures that the process does not get stuck in a local optimum. Our implementation uses a maximum of 100 generations. If this number is reached before homogeneity has occurred, the algorithm is stopped. This did, however, not happen in our simulations.

5.2 Travel time savings

It seems obvious that placing more supporting units in the scenario improves the performance of the dissemination process and hence higher travel time savings can be achieved. In Figure 6 this expectation is confirmed. On the x-axis the number of used supporting units is shown. The y-axis shows the relative travel time for the best supporting unit vector found by the genetic algorithm. The error bars show 99.9 % confidence intervals. A value of one means that no savings can be achieved compared to the current travel time on the optimal standard route, i. e., $\|r^*\| = \|r^*_{\text{std}}\|$. If supporting units are placed at all 100 considered locations, an average car needs a relative travel time of 0.9 compared to the standard travel time. This is equivalent to a travel time saving of 10 %.

Similar time reductions, however, are also possible with fewer supporting units. Even without any infrastructure support the aggregation based dissemination scheme is able to deliver data to cars that can help to improve their route. The knee in the plot indicates that a good tradeoff between cost and utility in the considered city could be between 10 and 30 SUs.

It should be noted that a large number of cars does not profit from the additional information, since the standard path to their destination is not congested, or despite a certain level of congestion no better alternative route exists. Those vehicles would not profit from any traffic information system at that time. However, they are included in the calculation of the average travel time savings. Cars for which better routes actually do exist often exhibit substantially larger improvements than the above average values.

This can be seen by investigating the distribution of travel time savings. Figure 7 shows the cumulative distribution function of the individual relative travel times. The large fraction of cars with a relative travel time of one includes all those cars that would choose the same path without any dynamic information.

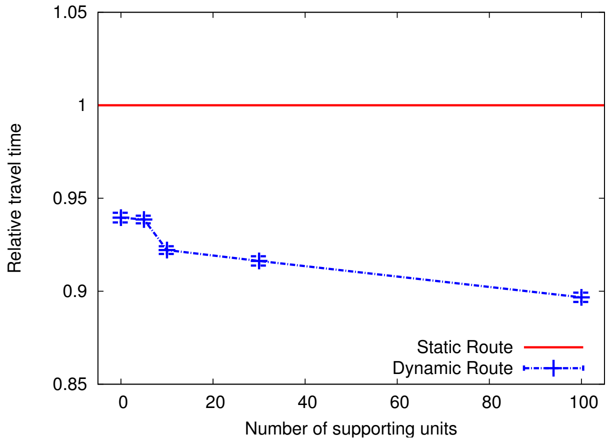


Figure 6: Performance evaluation of different active supporting units.

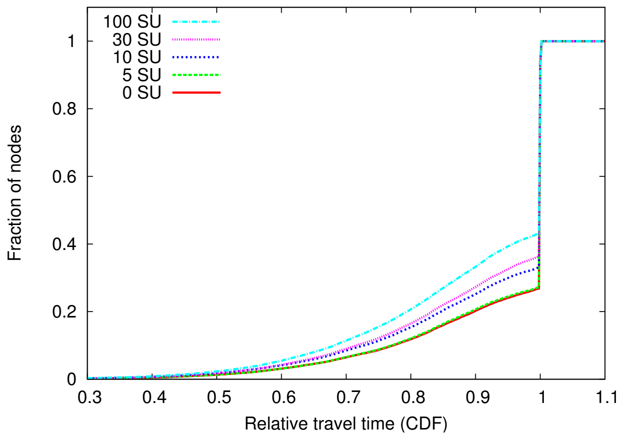


Figure 7: Cumulative distribution function of the relative travel times.

5.3 Genetic algorithm evolution

Figures 8 and 9 depict the evolution of the vector of supporting units while the genetic algorithm is running. On the x-axis the IDs of all possible SU location are shown. The z-axis, on the right hand side of the figures, shows the progression of generations. For each SU location in each generation, the z axis shows the number of individuals in which the respective location is occupied by an SU. Initially the SU vectors are chosen randomly, so in the first generations, SUs are distributed very homogeneously over the locations. After some generations, however, clear trends become visible and the individuals start to become more and more similar. Some locations are virtually completely abandoned quite early. At some point a specific combination of SU locations becomes predominant and the genetic algorithm cannot gain any further improvement.

A final evaluation of the results of the genetic algorithm is depicted in Figures 10 and 11. Here, the locations of the supporting

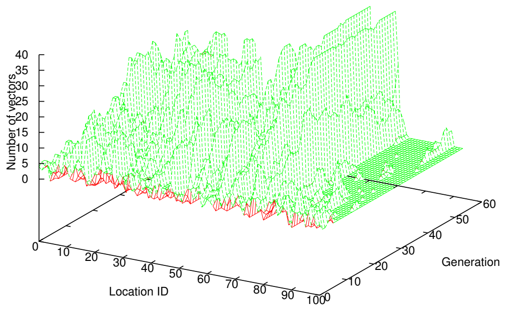


Figure 8: Evolution of SU vectors with ten active SUs.

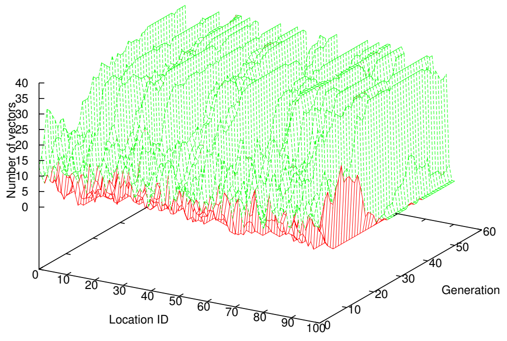


Figure 9: Evolution of SU vectors with thirty active SUs.

units are shown in the analyzed scenario. The crosses represent the 100 possible SU locations. The locations chosen by the genetic algorithm are marked using squares. It is conspicuous that SUs are distributed quite uniformly around the city center, in particular if few of them are available. When more supporting units are available, like in Figure 11, SUs are also placed within the city center.

6. CONCLUSION

In this paper we have presented an aggregation scheme for travel time data in road networks. In order to disseminate information within a large network, aggregation is done by means of a multi-layer hierarchy of approximations of the road network. A landmark based aggregation scheme distributes information about the travel times between prominent points of the road network in order to build an abstract view of more distant regions.

Given this aggregation scheme, it then becomes possible to tackle a second big issue in a VANET-based traffic information system: how and where infrastructure should be used in order to improve information dissemination over larger distances. We have introduced an approach for optimizing the placement of networked roadside infrastructure—supporting units—based on genetic algorithms. By a simulation methodology that separates movement and network issues from application behavior it becomes possible to estimate the travel time savings achieved by a given vector of active SU locations. These savings can be used as a fitness metric, making an

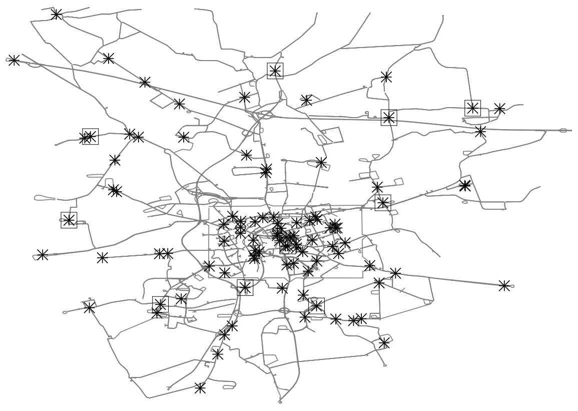


Figure 10: Location of ten active SUs.

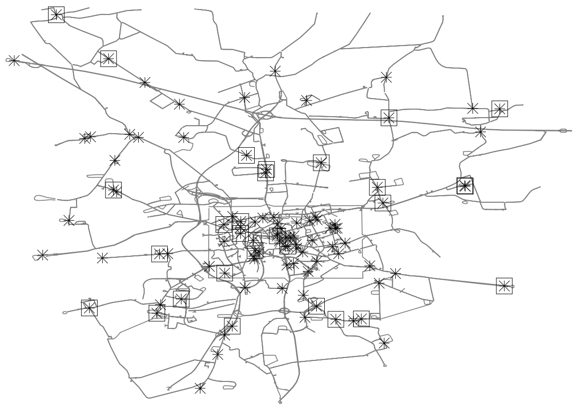


Figure 11: Location of thirty active SUs.

application-centric optimization approach feasible. We have confirmed the viability of this approach and assessed the achievable improvements by applying it to a large-scale city VANET model.

7. REFERENCES

[1] C. Adler, R. Eigner, C. Schroth, and M. Strassberger. Context-adaptive information dissemination in VANETs – maximizing the global benefit. In *CSN '06: Proceedings of the 5th IASTED International Conference on Communication Systems and Networks*, pages 7–12, Aug. 2006.

[2] M. Caliskan, D. Graupner, and M. Mauve. Decentralized discovery of free parking places. In *VANET '06: Proceedings of the 3rd ACM International Workshop on Vehicular Ad Hoc Networks*, pages 30–39, Sept. 2006.

[3] E. Cantú-Paz. A summary of research on parallel genetic algorithms. Technical Report 95007, University of Illinois at Urbana-Champaign, July 1995.

[4] P. Golle, D. Greene, and J. Staddon. Detecting and correcting malicious data in VANETs. In *VANET '04: Proceedings of the 1st ACM International Workshop on Vehicular Ad Hoc Networks*, pages 29–37, Oct. 2004.

[5] M. Jakobsson and S. Wetzel. Efficient attribute authentication with applications to ad hoc networks. In *VANET '04: Proceedings of the 1st ACM International Workshop on Vehicular Ad Hoc Networks*, pages 38–46, Oct. 2004.

[6] C. Lochert, B. Scheuermann, M. Caliskan, and M. Mauve. The feasibility of information dissemination in vehicular ad-hoc networks. In *WONS '07: Proceedings of the 4th Annual Conference on Wireless On-demand Network Systems and Services*, pages 92–99, Jan. 2007.

[7] C. Lochert, B. Scheuermann, and M. Mauve. Probabilistic aggregation for data dissemination in VANETs. In *VANET '07: Proceedings of the 4th ACM International Workshop on Vehicular Ad Hoc Networks*, pages 1–8, Sept. 2007.

[8] T. Nadeem, S. Dashtinezhad, C. Liao, and L. Iftode. TrafficView: Traffic data dissemination using car-to-car communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 8(3):6–19, July 2004.

[9] F. Picconi, N. Ravi, M. Gruteser, and L. Iftode. Probabilistic validation of aggregated data in vehicular ad-hoc networks. In *VANET '06: Proceedings of the 3rd ACM International Workshop on Vehicular Ad Hoc Networks*, pages 76–85, Sept. 2006.

[10] M. Raya, A. Aziz, and J.-P. Hubaux. Efficient secure aggregation in VANETs. In *VANET '06: Proceedings of the 3rd ACM International Workshop on Vehicular Ad Hoc Networks*, pages 67–75, Sept. 2006.

[11] S. N. Sivanandam and S. N. Deepa. *Introduction to Genetic Algorithms*. Springer Verlag, 2007.

[12] L. Wischhof, A. Ebner, and H. Rohling. Information dissemination in self-organizing intervehicle networks. *IEEE Transactions on Intelligent Transportation Systems*, 6(1):90–101, Mar. 2005.