Coverage and Mobile Sensor Placement for Vehicles on Predetermined Routes: A Greedy Heuristic Approach

Junade Ali and Vladimir Dyo
Department of Computer Science and Technology, University of Bedfordshire, Luton, U.K.

Keywords: Mobile Sensors, Intelligent Transportation Systems, Smart Cities, Optimal Route Selection, Set Cover Problem.

Abstract: Road potholes are not only nuisance but can also damage vehicles and pose serious safety risks for drivers. Recently, a number of approaches have been developed for automatic pothole detection using equipment such as accelerometers, image sensors or LIDARs. Mounted on vehicles, such as taxis or buses, the sensors can automatically detect potholes as the vehicles carry out their normal operation. While prior work focused on improving the performance of a standalone device, it simply assumed that the sensors would be installed on the entire fleet of vehicles. When the number of sensors is limited it is important to select an optimal set of vehicles to make sure that they do not cover similar routes in order to maximize the total coverage of roads inspected by sensors. The paper investigates this problem for vehicles that follow pre-determined routes, formulates it as a linear optimization problem and proposes a solution based on a greedy heuristic. The proposed approach has been tested on an official London bus route dataset containing 713 routes and showed up to 78% improvement compared to a random sensor placement selected as a baseline algorithm.

1 INTRODUCTION

Road surface condition monitoring is essential for maintaining a safe and efficient public transportation. Road defects such as potholes cause accidents, damage vehicles and slow down traffic, and for this reason, municipalities invest millions to maintain and repair the roads. However, timely detection of potholes remains problematic due to very high cost of inspection vehicles and the massive total length of the roads. Recently, a number of approaches have been developed, where mobile wireless sensors deployed on taxis, buses or utility vehicles, automatically inspect the road as the vehicles carry their normal operation, opening a way for a large scale and autonomous road monitoring. While prior work focused on improving the sensor detection performance, data collection and energy efficiency, it was largely assumed that the set of vehicles on which sensors would be deployed is either random (in case of taxis), pre-determined (known) or that the sensors would be deployed on the entire fleet. The problem arises when a limited number of sensors needs to be *optimally* deployed on vehicles that typically follow pre-determined routes, such as buses or utility vehicles, because placing sensors on the vehicles that follow similar routes would result in sub-optimal road coverage.

The problem of coverage for mobile sensors has been explored in the context of path planning for patrolling applications (Murray, 2016), persistent monitoring tasks for autonomous vehicles (Kuhlman et al., 2014) and data collection (Di Francesco et al., 2011). In particular, (Kuhlman et al., 2014) investigates the optimal mobile coverage problem in the context of unmanned vehicles, and proposes an automated path planner that minimizes vehicle's path length whilst maximizing the information gathered along that path. The method however was designed for a single unmanned vehicle and does not scale to multiple vehicles. (Contreras et al., 2016) tackles an optimal sensor placement problem for sensor networks deployed on highway segments to maximize information collected while minimizing monetary cost. This work is similar in spirit to the proposed approach but targets fixed rather than mobile sensors. To the best of our knowledge, no previous work considers an optimal placement of mobile sensors on vehicles that follow pre-determined routes.

In this paper we present a novel approach for an optimal mobile sensor placement that maximises the total length of the road inspected by sensors. The approach starts by representing each vehicle route as a

Ali, J. and Dyo, V.
Coverage and Mobile Sensor Placement for Vehicles on Predetermined Routes: A Greedy Heuristic Approach.
DOI: 10.5220/0006469800830088
In *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017) - Volume 6: WINSYS*, pages 83-88
ISBN: 978-989-758-261-5

set of road segments, and using set cover theory to formulate the problem as a linear optimisation problem that maximises the total number of segments for a given number of vehicle routes. A greedy heuristic algorithm is then proposed, which solves the problem in a computationally efficient way. The implementation of the algorithm is not trivial as it requires an efficient method for subtracting geographical routes, for which a simple solution is proposed. The approach has been implemented in Go programming language, and evaluated on the official Transport for London (TfL) bus dataset. The simulations have been conducted to evaluate the algorithm performance for different number of selected routes. Based on the simulation results, we found that the proposed approach provides a significant improvement over a random route selection selected as a baseline algorithm.

2 RELATED WORK

The traditional pothole detection relies on manual road surface inspection or automated inspection using specialised vehicles. Because of high cost and labour-intensive process, a typical survey is only done once in 4 years (Zhang et al., 2014) whereas potholes can develop quickly, with little or no advance warning.

Recently, the rapid advances in low cost wireless sensors, open source hardware platforms and sensing technologies has led to a surge of projects on low cost pothole detection systems in both academia and industry. A number of such systems have been designed, deployed and evaluated by researchers around the world (Mohan et al., 2008) (Eriksson et al., 2008). The majority of low cost systems are vibration-based, and use an accelerometer to detect a jolt, which happens when a vehicle drives over a pothole. The detection accuracy depends on multiple factors including vehicle type, pothole location, road conditions and driver behaviour. The approach proposed by (Eriksson et al., 2008) was able to correctly detect 88.9%-92.4% of potholes in an experiment with hand-labelled training data and loosely labelled data respectively. It should be noted that the system mounted on utility vehicles can repetitively check the same road segment over and over again, so may not require very high detection rate. This is in contrast to dedicated pothole monitoring systems, which are designed to reliably detect potholes through a single passing of the road and therefore optimised for high detection accuracy. (Hull et al., 2006) introduces a mobile sensor computing system known as CarTel, this system allows data consumers to issue SQL queries to assess data from CarTel devices.

Most deployments relied on random deployment on either taxi cars or on specially selected cars. To the best of our knowledge, none of the works considers the problem of mobile sensor placement under the scenario, where sensors need to be deployed on vehicles that run along specific routes, such as garbage trucks, buses or other utility vehicles.

At the same, there have been a number of works on route planning in the public transportation networks, such as optimal journey planning depending on passenger and service preferences (Botea et al., 2013), trust-oriented trip planning, app-based taxi pooling (Chen, 2014) and public vehicles tracking (Zhu et al., 2016). None of these works considers a problem of optimal route selection for vehicles equipped with mobile sensors. (Contreras et al., 2016) tackles an optimal sensor placement problem for sensor networks deployed on highway segments to maximize information collected while minimizing the monetary cost. Our work is similar in spirit, but targets mobile rather than fixed sensors.

Finally, some related work exists in the area of mobile sensor networks but focuses mostly on data collection algorithms, energy management (Di Francesco et al., 2011) and path planning for patrolling applications (Murray, 2016). The recent works on mobile video surveillance and ubiquitous video acquisition are also relevant, but they focus mostly on improving image quality under constrained resources, image processing and analysis issues, without considering a problem of optimal sensor placement to maximize the coverage.

3 OPTIMAL ROUTE SELECTION

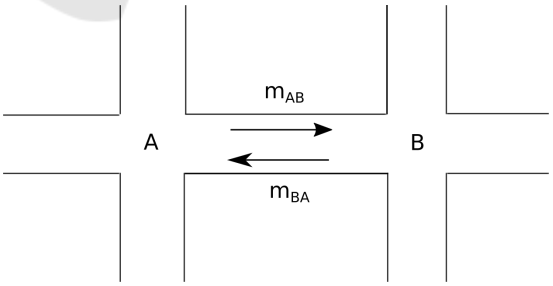

Figure 1: Road segment representation.

We represent each bus route as a set of road segments, $R_i = \{m_1, m_2, m_3 ... m_n\}$, where each segment m_i represents part of the road between two adjacent intersections and which can belong to more than one bus route. The segments are directed, so a two-way street would be represented by two segments as shown on Fig. 1. The problem is formulated as follows: given a

collection of sets $S = \{R_1, R_2, ...R_n\}$ and an integer k, find a subset S' of S such that the number of covered elements is maximized, and $|S'| <= k$, i.e. select a set of k vehicle routes that maximizes the total number of selected segments. The vehicle route selection problem can be expressed as an integer program:

$$maximize \sum R_i$$

$$s.t. \sum_{i=1}^{N_{routes}} u_i = k$$

$$u_i = \{0,1\},\ i = 1,2,3...N_{routes}$$

Where u_i is a decision variable showing whether a route R_i has been selected, k is the number of routes to select, N_{routes} is the total number of routes.

3.1 Greedy Approximation

The formulated problem is an instance of a known problem in combinatorics called a maximum coverage problem, which is known to be NP-complete (Hochbaum, 1997), meaning that an optimal solution requires analyzing every possible combination of routes. This can be computationally very hard for large number of routes. However, a number of approximations exist, with the best known being in $\theta(log\, n)$. The greedy algorithm is known as the best-possible polynomial time algorithm for maximum coverage problem and works by iteratively selecting a set R_i that contains the largest number of elements (road segments), and then removing selected elements from all the remaining sets in the collection. The process repeats until K sets are selected. The proof of correctness is not described here, but can be found in (Hochbaum, 1997).

However, the greedy algorithm's limitation is that it assumes that all road segments are of equal length, resulting in a route with 2 short road segments being preferred over a route with 1 long road segment. Overcoming this limitation requires redefining a road segment as a segment with a unit length, so that the number of segments is proportional to route distance. This way, a single long road segment can be represented as multiple unit length elements, and will be preferred over shorter road segments without modifications to an algorithm. The pseudocode based on a direct translation of a classic greedy algorithm for a route selection problem is shown in Algorithm 1. The proposed algorithm is computationally lightweight and requires $O(N^2)$ operations.

Data: $\{R_1, R_2, ...R_{N_{routes}}\}$ - a set of bus routes
$R_i = \{m_1, m_2, ...m_{n_i}\}$ - each vehicle route as a set of road segments
N_{routes} - total number of vehicle routes
k - number of routes to select
Result: *MaximumRouteSet* - a set of route IDs that maximizes coverage
initialization;
$MaximumRouteSet := \emptyset$;
repeat
  // Route retrieval: select the route that has the largest number of uncovered elements
  $i := argmax(UncoveredLength(R_i))$
  $MaximumRouteSet := MaximumRouteSet + i$
  // Route Exclusion: Subtract the covered road segments from the remaining routes. Note that an efficient implementation can simply 'mark' the covered road segments without actual 'subtraction'
  for $j \in RemainingRoutesIDs$ **do**
    $R_j := R_j$ - R_i
  end
until $length(MaximumRouteSet) = k$;

Algorithm 1: Maximum route coverage.

4 EVALUATION

4.1 Data Set

The algorithm is evaluated on Transport for London dataset (TFL, 2016) containing 713 bus routes in Greater London, released for application developers to create apps such as locating nearest bus stops or perform journey planning. The data is available as a CSV file, where each record specifies the coordinates of a bus stop (*Location_Easting* and *Location_Northing*), a bus route (*Route*), the sequence number along the route (*Sequence*), the name and the code for the bus stop (*Stop_Name* and *Bus_Stop_Code*), and the direction of travel (*Run*). The dataset contains routes for a total of 19801 different bus stops. The summary of data format is described in Table 1. The actual street by street path for each bus route has been extrapolated from bus stop coordinates by using Google's turn-by-turn navigation API (Wallace et al., 2014). Here, an assumption has been made that a bus moves along the shortest path between consecutive bus stops, which works well in most cases.

The processed data set contains GPS coordinates of the turn-by-turn road segments for each bus route. The need to run the bus routes through a turn-by-turn

Figure 2: The original dataset contains bus stop locations only, which is not sufficient for running a route selection algorithm.

Figure 3: The processed dataset contains a more detailed route information obtained through turn-by-turn direction API.

direction API is visualized on Fig. 2 and Fig. 3:

The implementation was done in the Go programming language known for its support for concurrent operations (Donovan and Kernighan, 2015). The script iterates through all bus stops for each bus route, extracts the coordinates of all road intersections and saves the data in a KML file as well as MySQL database. KML was used for visualizing the routes in Google Earth, while MySQL was used for further processing as described in the next subsection. Visualizing KML data in Google Earth proved to be beneficial for discovering defects in the data supplied. For example, the Ordinance Survey Easting co-ordinate for a stop on Summit Close contained an error which affected three bus routes. This was reported to TfL, who corrected the co-ordinate and updated their dataset accordingly. Whenever an individual bus route was processed, it was processed concurrently with other bus routes. This was advantageous in the fact that larger bus routes with substantial API calls did not block the processing of other routes. The script processed 10 routes concurrently on a laptop with a dual-core 2.7 GHz Intel i5 with 8GB RAM, and converted the entire data set in 20 minutes.

4.2 Geographical Route Exclusion

Recall that the optimization algorithm works by iteratively repeating longest remaining route retrieval and route exclusion procedures. Both procedures can be implemented trivially, if all road segments were represented as abstract set elements with unique numerical IDs. The actual data for each bus route comprises a set of road segments defined by their GPS coordinates as illustrated on Fig. 4 where white lines indicate several bus routes going around the same roundabout. Note that each bus route uses its own set of road segments (white lines) to describe the same road around the roundabout, which complicates route exclusion. For route exclusion to work correctly, removing any bus route going through the roundabout should automatically remove all nearby road segments that belong to other bus routes in the same area.

We propose the following simple implementation for geographical route retrieval and route exclusion. Let A = $(longtitude1, latitude1)$ and B = $(longtitude2, latitude2)$ be the co-ordinates of the start and end points of the selected road segment. The proposed route exclusion algorithm works by placing a rectangle with $AA'BB'$ on a map, where $A' = (longtitude1, latitude2)$ and $B' = (longtitude2, latitude1)$, and then removing all road segments that start and end within the rectangle, as shown in Fig 5. In the diagram below the white lines indicate a bus route, with the grey rectangles indicating the rectangles $AA'BB'$, any other route segments which touch or overlap the red squares will be removed. This allows us to ensure that segments that run parallel (but would not ordinary overlap), can still be identified and removed alongside the target segment.

The distances between each co-ordinate are unequal, however in practice as each small turn on the route creates a new segment, the route segments are therefore small enough not to cause a problem (no segment was greater than 100 meters apart). If there is a particularly long segment running alongside a road which has another road running parallel alongside side it at limited proximity, it is possible that a segment on the other road may be removed. Therefore, it is vital that segment lengths are kept to a minimum in order to avoid this behaviour. Our implementation is able to identify long segments in the dataset and split them into smaller segments by modifying the end point of a segment, and inserting a new segment. An alternative method of removing nearby road segments would have been by identifying points which are within a given radius of a set of midpoints of the given segment. However, this approach would require

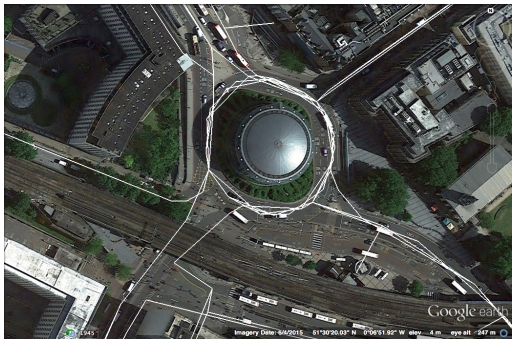

Figure 4: Charlie Chaplin Walk roundabout.

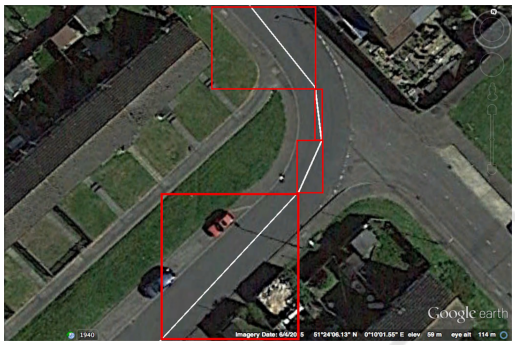

Figure 5: Route Exclusion problem.

the knowledge of which road a given segment is on to validate the fact that two segments are indeed on the same road.

The implementation process starts off by populating a MySQL database with an entry for each road segment. The length of each route is calculated dynamically as a total length of road segments on that route. By using a function that calculates the distance between co-ordinates we can calculate the total length of the routes and mark all segments of the largest route as deleted. As we iterate through each part of the route to mark it as a deleted, we accordingly mark any other route segments on the remaining routes as deleted if the segments' co-ordinates are within the start and end points of the route segment being deleted as described below. As the next route that is removed depends on the previous route removed, it was not possible to concurrently process multiple routes together. We did not however feel that this was a limiting factor as this step took a matter of seconds.

5 RESULTS

The goal of the evaluation is to compare the performance of the proposed greedy algorithm with a baseline method, where the routes are selected randomly. The baseline algorithm was run three times and the average result was taken, whereas the Greedy algorithm only needs to be run once, as it is deterministic and always selects the same set of routes regardless of how many times the simulation is run.

Fig. 6 shows the total route coverage in meters depending on the number of selected routes for both proposed greedy approximation and baseline algorithms. The total route coverage increases linearly with the number of selected routes for a baseline approach. It can be seen that for each number of selected routes, the proposed algorithm provides higher route coverage. At just 100 selected routes (around 14% of routes), greedy algorithm covers 25% of the total road length compared to 14% achieved by a baseline algorithm, which corresponds to 78% improvement. The greedy algorithm provides a higher coverage because it attempts to select the longest route each time, and then excludes the relevant segments from the remaining routes. This improvement in coverage decreases to 54%, 38% and 12% for 200, 300 and 600 routes respectively. Whilst the improvement reduces with the number of selected routes, the result clearly demonstrates that at any point, greedy approximation performs better than a baseline. The results can be particularly useful to demonstrate an effectiveness of an approach when not all the vehicles can be equipped with pothole sensors. This can be the case for deployments in large geographical areas that consist of thousands of routes.

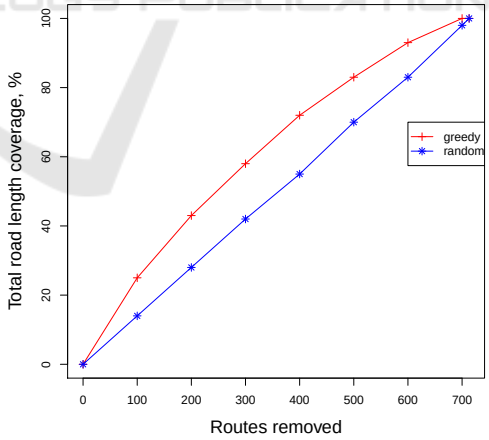

Figure 6: Total route coverage against the number of selected routes.

6 CONCLUSIONS

Low cost mobile sensors provide a promising solution for a large scale and timely pothole detection. While a number of solutions have been proposed recently, most focus on sensor design, signal processing and

data collection ignoring an important question of sensor placements. We have proposed a novel approach for selecting an optimal set of vehicles that maximizes the geographical road coverage. We have formalised the problem as a maximum coverage problem and proposed a greedy heuristic, which was evaluated on TfL London bus route data set. The evaluation has shown up to 78% improvement over a random route selection selected as a baseline algorithm. Some applications may find it useful to assign priorities to certain roads depending on the road traffic or road size. For a pothole detection application, it may be useful to have a higher scanning frequency for strategic and secondary distributor roads and lower frequency for local access and link roads. This can be modelled as a weighted maximum coverage problem, the evaluation of which we leave for future work.

REFERENCES

(2016). London Datastore. Data.london.gov.uk. https://data.london.gov.uk/dataset/tfl-bus-stop-locations-and-routes. [Online; accessed 16-Oct-2016].

Botea, A., Nikolova, E., and Berlingerio, M. (2013). Multi-modal journey planning in the presence of uncertainty.

Chen, W. (2014). Technical Improvements on Mobile App Based Taxi Dispatching System. http://www.atlantis-press.com/php/paper-details.php?id=12704.

Contreras, S., Kachroo, P., and Agarwal, S. (2016). Observability and sensor placement problem on highway segments: A traffic dynamics-based approach. *IEEE Transactions on Intelligent Transportation Systems*, 17(3):848–858.

Di Francesco, M., Das, S. K., and Anastasi, G. (2011). Data collection in wireless sensor networks with mobile elements: A survey. *ACM Trans. Sen. Netw.*, 8(1):7:1–7:31.

Donovan, A. A. and Kernighan, B. W. (2015). *The Go Programming Language*. Addison-Wesley Professional, 1st edition.

Eriksson, J., Girod, L., Hull, B., Newton, R., Madden, S., and Balakrishnan, H. (2008). The pothole patrol: Using a mobile sensor network for road surface monitoring. In *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, MobiSys '08, pages 29–39, New York, NY, USA. ACM.

Hochbaum, D. (1997). Approximation algorithms for np-hard problems, 1st ed. boston. *PWS*, pages 94–143.

Hull, B., Bychkovsky, V., Zhang, Y., Chen, K., Goraczko, M., Miu, A., Shih, E., Balakrishnan, H., and Madden, S. (2006). Cartel: A distributed mobile sensor computing system. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, SenSys '06, pages 125–138, New York, NY, USA. ACM.

Kuhlman, M. J., vec, P., Kaipa, K. N., Sofge, D., and Gupta, S. K. (2014). Physics-aware informative coverage planning for autonomous vehicles. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4741–4746.

Mohan, P., Padmanabhan, V. N., and Ramjee, R. (2008). Nericell: Rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, pages 323–336, New York, NY, USA. ACM.

Murray, A. T. (2016). Maximal coverage location problem. *International Regional Science Review*, 39(1):5–27.

Wallace, B., Goubran, R., and Knoefel, F. (2014). Measurement of signal use and vehicle turns as indication of driver cognition. In *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 3747–3750.

Zhang, Z., Ai, X., Chan, C., and Dahnoun, N. (2014). An efficient algorithm for pothole detection using stereo vision. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 564–568.

Zhu, M., Liu, X. Y., Tang, F., Qiu, M., Shen, R., Shu, W., and Wu, M. Y. (2016). Public vehicles for future urban transportation. *IEEE Transactions on Intelligent Transportation Systems*, 17(12):3344–3353.