Protecting Location Privacy Through Path Confusion

Baik Hoh
WINLAB
ECE Department
Rutgers, The State University of New Jersey
Email: baikhoh@winlab.rutgers.edu

Marco Gruteser
WINLAB
ECE Department
Rutgers, The State University of New Jersey
Email: gruteser@winlab.rutgers.edu

Abstract

We present a path perturbation algorithm which can maximize users' location privacy given a quality of service constraint. This work concentrates on a class of applications that continuously collect location samples from a large group of users, where just removing user identifiers from all samples is insufficient because an adversary could use trajectory information to track paths and follow users' footsteps home.

The key idea underlying the perturbation algorithm is to cross paths in areas where at least two users meet. This increases the chances that an adversary would confuse the paths of different users. We first formulate this privacy problem as a constrained optimization problem and then develop heuristics for an efficient privacy algorithm. Using simulations with randomized movement models we verify that the algorithm improves privacy while minimizing the perturbation of location samples.

1 Introduction

The continuous improvements in accuracy and cost of Global Positioning System (GPS) receivers are driving new location-based applications. Cellular communication technology as well as GPS can provide users' location information within 100 meters for 66 percent and 300 meters for 95 percent of the calls, which is mandated by the Federal Communications Commission for E911. The automotive industry intends to use vehicles as a mobile sensor platform for collecting information about traffic jams, weather, and road conditions [25]. Automatically collecting location information is also useful for a governmental Department of Transportation (DOT), which can use Origination-Destination (OD) statistics from many users for traffic analysis [5].

Sharing location information for such applications, however, raises privacy concerns. For example, in the United States, the "Location Privacy Protection Act of 2001" [1] and "Wireless Privacy Protection Act of 2003" [2] designate that an individual's location data can be used without prior agreement only for purposes that enhance public welfare. One possible technical solution is that individual location data is processed on a trusted computing device and only aggregated data is distributed to other parties. This may not always be feasible if the aggregation function is too complex or requires inputs that are not available on the device. Another common solution is anonymization via removal of identifiers. In the case of location information, however, information on users' trajectories enables an adversary to follow users' footsteps because there exists a high spatial correlation between successive location samples. Multi Target Tracking (MTT) algorithms [21] are a well-studied technique to link subsequent location samples to individual users who periodically report anonymized location information. Thus, naive anonymization cannot solve the location privacy problem for path information [14].

In this work, we conduct a feasibility study on mechanisms that prevent an adversary from tracking a complete individual path. Perturbation algorithms which impose tolerable errors on original location samples to maintain user-specified levels of quality-of-service. We approach the development of such an algorithm in four steps. First, we define the model for location privacy in terms of confidence and spatial distance. Second, we define quality of service (QoS) in terms of the error that the algorithm imposes on location samples. Third, we derive an algorithm from a constrained optimization problem formulation, which maximizes the metric of location privacy given a QoS requirement. Fourth, we calibrate our algorithm for an automotive traffic monitoring system and present simulation results with random movement models. We limit our discussion to applications which receive periodic and anonymous location samples. In addition, we concentrate on applications where data can be processed offline but our results are also applicable to online applications that tolerate slight delays.

The remainder of this paper is structured as follows. Sec-

tion 2 defines the class of applications and the privacy problem that this paper addresses. We describe how an adversary could use MTT algorithms to form paths from anonymous location samples. Section 3 defines a mathematical privacy model and describes a numerical algorithm to the path perturbation problem. We apply our algorithm to random movement models and compare its performance and limitations to a random perturbation baseline algorithm in section 4. In section 5, we discuss the implications of our results for GPS-based application by comparing our simulation environment with a real world scenario. We compare with related work in section 6 and discuss future works before we conclude.

2 Threat assessment and Multi Target Tracking

We motivate the class of applications considered in this paper with two examples from the automotive telematics domain[1]. One application is traffic monitoring to provide drivers with quicker feedback on road conditions. Selected vehicles could periodically send their locations, speeds, road temperatures, windshield wiper status and other information to a traffic monitoring facility. This data reveals the length of traffic jams (through speed and position), weather condition such as rain (through windshield wiper activity), and slick road conditions (through frequent anti-lock breaking). Using vehicles as mobile sensing platforms promises dramatic cost reductions over deploying specialized roadside sensors. The GuideStar project [20], for example, plans to implement such a system.

Another application is transportation planning. Periodically, a DOT may asks for aggregated data across a large number of users. The DOT can infer road usage from the distribution of cars and calculate each car's speed from successive location samples. These average speeds provide information on the frequency of traffic jams and average travel time on specific roads. These kind of inferred statistics could be used for traffic light scheduling, road redesign, and other transportation optimizations.

2.1 System Model

Although these applications differ in their detailed data requirements, they fit a common system model. Table 1 lists different data requirements for the traffic monitoring and traffic planning applications. For example, to estimate the average velocity on a road segment, it is sufficient to track individual vehicles for the length of the road segment. For transportation planning, it is necessary to know origin and destination of a vehicle to determine which alternate routes a vehicle could take. In the traffic monitoring case, the driver is the data consumer, thus the driver can define the level of accuracy needed. For transportation planning, that decision must lie with government agencies. Alleviating traffic jams also requires immediate feedback, that can only be provided by an online application. For transportation planning an offline, batch processing application may be sufficient. In general, however, both applications require access to periodic location samples from a large number of vehicles and do not need to receive identity information with the location data.

	Location-Based Applications	
	Traffic Monitoring	Transportation Planning
User Density (N/km^2)	4	4
Timeliness	Online	Offline
Data Requirements	Road segment tracking	O-D Tracking
Data Accuracy	User-configurable	Predefined

Table 1. Characteristics of Location-Based Applications

We assume a system model that interposes a proxy location server between applications and vehicles (this proxy could be operated by a cellular or telematics service providers with whom vehicle owners have a service agreement). The proxy provides applications with access to location samples. It anonymizes all traces before passing data on by removing identifiers such as user ids or network addresses from the data.

We assume that the positioning, cellular communications, and proxy infrastructure are appropriately secured and trustworthy. Location information should be encrypted when transmitted between vehicles and proxy and the proxy can use the mix concept before passing data on to applications. The proxy itself could be secured through a combination of contractual obligations, privacy legislation, and secure hard- and software systems.[2]

2.2 Threats

The privacy problem considered here is data protection after transmission to third-party application service

[1] Efforts such automated road toll collection and taxation are likely to provide GPS and communication infrastructure in vehicles that could be reused for the following applications. For example, the State of California considers taxing drivers by the mile. This would be implemented through an in-car GPS receiver that keeps track of its mileage. When a driver visit a gas station, the vehicle could communicate the location log files to a computer in the gas pump which automatically adds the new tax to the bill [16].

[2] Cell phone operators already have access to and need to protect users' position information. Coarse information is available via the GSM Home Location Register, for example, and detailed information can be provided through positioning technologies deployed under the E911/E112 programs.

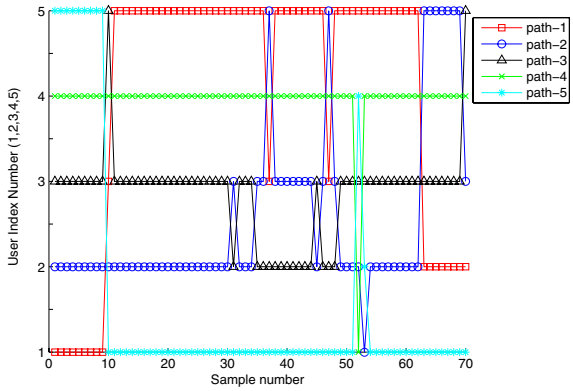

Figure 1. Disambiguation of paths. The five curves represent the association of location sample paths to users by an adversary. For example, path-4 is wholly assigned to user 4 except its 53rd point assigned to user 1.

providers, with whom a user may have no direct business relationship and may not be able to choose between alternate providers. Although the location proxy forwards only anonymous location samples, such naive anonymization is not sufficient because location traces are often so distinctive that users can be reidentified. A trace may begin on a suburban home's driveway, for example, which allows household identification through correlation with a geocoded household address database. Similarly, Beresford and Stajano [6] mention that the location traces collected in an office environment through the Active Bat system could be correctly reidentified by knowing the desk positions of all workers and correlating them with the traces. The spatial and temporal correlation between successive location samples creates challenges even when every location sample is anonymized individually. In this case the adversary has no information about which subset of samples belongs to a single user. Multi-target tracking (MTT) algorithms developed in the tracking systems community can however recreate the most likely paths based on general assumptions about user's movements.

To illustrate the performance of MTT algorithms, we have applied a simplified version of Reid's multiple hypotheses tracking algorithm [21] to a sample of five GPS paths collected by students on a college campus. Multiple hypotheses tracking, based on Kalman filtering, is a basic work in the field. Most paths could be trivially distinguished based on time or because of large distances between them. Thus, we overlaid the paths to create a more challenging synthetic scenario. Figure 1 shows the MTT output for this scenario. The five curves show the paths that the algorithm reconstructed from the anonymous samples. A step in value of a curve means that the algorithm has misassigned samples to a wrong user—five constant lines would mean perfect reconstruction. The algorithm clearly confuses a number of sample points, but it tracks users two and four for almost the entire scenario, and others for an extended period of time.

We pose the research problem as increasing the level of confusion while still enabling statistical location-based applications. Longer tracking durations lead to an accumulation of information that eventually will lead to identification of a user. Ideally, a privacy mechanism would limit the duration over which users can be tracked, so that all users can enjoy a similar level of privacy and are not depended on traffic densities in different areas or other factors beyond their control. In this discussion, we are most concerned with attacks that can be easily automated and applied to large numbers of users rather than preventing tedious detective work targeted at a single user.

3 Disclosure Control Algorithm

In this section, we present a formal metric for location privacy and derive a disclosure control algorithm which addresses the trade-off between location privacy and quality of service.

3.1 Location Privacy and Quality of Service Metrics

Intuitively, we define the degree of location privacy as the accuracy with which an untrusted party can locate an individual user. A location privacy metric has to take into account distance and uncertainty. Privacy is intrinsically related to the concept of uncertainty, thus entropy-based metrics have been used to evaluate privacy in anonymous communication and data mining systems (e.g., [22, 11, 3]). Entropy also proves useful in evaluating location anonymity. It is typically defined as $H(k) = -\sum_{i=1}^{I} p_i \log p_i$ where, in the case of location privacy, the p_i describe the adversaries probabilities for different assignments of user identities to the observed positions and I indicates the total number of such assignment hypothesis. Entropy, however, does not consider whether the locations of these users are actually different. Consider an extreme case where two users, Alice and Bob, meet and report two anonymous location samples l_1 and l_2 to the location-based service. Assuming that adversaries have no additional information, they could not distinguish whether Alice has sent sample l_1 or l_2. Therefore entropy would report a high degree of anonymity. From an information privacy perspective, however, the uncertainty in the assignment does not matter because both possible assignments carry the same information. Privacy would only

be increased if l_1 and l_2 described different positions.

We choose an alternate metric, expectation of distance error, which captures how accurate an adversary can estimate a user's position. We define the expectation of distance error for a path as

$$E[d] = \frac{1}{NK} \sum_{k=1}^{K} \sum_{i=1}^{I} p_i(k) d_i(k) \tag{1}$$

where the d_i represent the total distance error between the correct assignment hypothesis and the hypothesis i. N denotes the number of users and K is the total observation time.

The data quality that location-based applications provide depends largely on the accuracy of location information. For example, a traffic monitoring application needs to map vehicle positions to road segments. Inaccurate location information may lead to misassignments. We select a general error-based metric because applications exhibit varying levels of robustness against location inaccuracies, which is difficult to capture in a single metric.

We define the mean location error (QoS) for a set of N different users' paths of length K as

$$QoS = \frac{1}{NK} \sum_{n=1}^{N} \sum_{k=1}^{K} \sqrt{(\widetilde{x_n}(k) - x_n(k))^2 + (\widetilde{y_n}(k) - y_n(k))^2} \tag{2}$$

where $x_n(k), y_n(k)$ is the actual and $\widetilde{x_n}(k), \widetilde{y_n}(k)$ the observed location of user n at step k. For simplicity, we have not considered the time dimension, but this definition can be easily expanded.

3.2 Path Confusion as an Constrained Nonlinear Optimization Problem

The key idea underlying the following privacy algorithm is the concept of path confusion. Every time two users' paths meet (we define meeting as being in close proximity) there is a chance for the adversary to confuse the tracks and follow the wrong user. A privacy algorithm can exploit this by perturbing location information in such meeting areas to increase the chances of confusion.

Considering one particular meeting area, we can formulate perturbation as a constrained nonlinear optimization problem. First, we form a cost function with the expectation of distance error $E[d(k)]$ at time k. We then add an inequality constraint on location variables, $\widetilde{x_n}(k), \widetilde{y_n}(k)$ for every user n at time k as follows.:

$$(\widetilde{x_n}(k) - x_n(k))^2 + (\widetilde{y_n}(k) - y_n(k))^2 \leq R^2 \tag{3}$$

where R is a user- or application-specific input parameter that defines the maximum permissible perturbation. The objective is to maximize

$$\max_{\forall \widetilde{x_n}(k), \forall \widetilde{y_n}(k)} \frac{1}{N} \sum_{i=1}^{I} p_i(k) d_i(k) \tag{4}$$

where the total distance error $d_i(k)$ and the adversary's probability $p_i(k)$ are described by the following equations.

$$d_i(k) = \sum_{n=1}^{N} \sqrt{(\widetilde{x_{m_i(n)}}(k) - x_n(k))^2 + (\widetilde{y_{m_i(n)}}(k) - y_n(k))^2} \tag{5}$$

$$p_i(k) \equiv P(\Omega_i^k \mid Z^k) \approx \prod_{n=1}^{N} f_n(\widetilde{x_{m_i(n)}}(k), \widetilde{y_{m_i(n)}}(k)) \tag{6}$$

The formulas to estimate the adversaries probability assignment are derived from Reid's Multi-Hypothesis Tracking algorithm [21]. The probability, $p_i(k)$ denotes the probability of hypothesis Ω_i [3] at time k, given the set of observations Z^k [4]. In equation 6, m_i is an assignment vector for the ith hypothesis. Each field j in the vector contains the user index that this hypothesis assigns to the j location sample.[5] Function f_n is a multivariate gaussian density described by the following equation:

$$f(z^k \mid \bar{x}^k) = N(z^k - H\bar{x}^k, B), \tag{7}$$

where x^k is the state vector consisting of estimated position and velocity at step k and z^k is a new observation vector. Here, $N(m, C)$ refers to a multivariate normal distribution

$$N(m, C) = e^{-\frac{1}{2} m^T C^{-1} m} / \sqrt{(2\pi)^n \mid C \mid}.$$

The state vector, x^k can be predicted from the previous state vector x^{k-1} according to a process model and z^k relates to the actual state through an observation model

$$x^k = Fx^{k-1} + w \qquad \text{and} \qquad z^k = Hx^k + v, \tag{8}$$

where w represents the process noise vector and matrix F describes a linear prediction of the next state given the previous state. Matrix H converts a state vector into the measurement domain and v represents the measurement noise vector. This linear Kalman model assumes that the process noise and the measurement noise are independent of each other and normally distributed with covariance matrices Q and S, respectively. To calculate $B = H\bar{P}^k H^T + S$

[3] It denotes ith hypothesis of the set of all hypotheses at time k which associate the cumulative set of location samples Z^k with N users. We may view Ω_i^k as the joint hypothesis formed from the prior hypothesis Ω_g^{k-1} and the association hypothesis for the current data set $Z(k)$.

[4] It denotes the cumulative set of location samples up through time k whereas $Z(k)$ indicates the set of location samples only at time k.

[5] All vectors together represent all permutations of users. Therefore, $(\widetilde{x_{m_i(n)}}(k), \widetilde{y_{m_i(n)}}(k))$ represents the observed position according to the ith hypothesis for user n.

and $N(z^k - H\bar{x}^k, B)$, we need to know both $\bar{x}^k$ and $\bar{P}$, which are calculated using the time update equation at the prediction step of Reid's multiple hypothesis tracking algorithm [21].

At each time step, the filter predicts the new target position as

$$\bar{x}^{k+1} = F\hat{x}^k \qquad \text{and} \qquad \bar{P}^{k+1} = F\hat{P}^k F^T + Q^T, \quad (9)$$

where $\hat{x}$ and $\hat{P}$ are the estimates after the last sample was received. (section 3 in Gruteser and Hoh's work [14] for more details).

We build the constrained optimization problem by using the equation 4 and the equation 3 of each user as a cost function and constraint equations, respectively. This optimization problem can be solved through a numerical approach, such as Sequential Quadratic Programming. For our experiments we relied on MATLAB's *fmincon* function.

With the set of the solutions $(\widetilde{x_n}(k), \widetilde{y_n}(k))$ for all users at time k to the optimization problem, the state vector for each user will be updated with the Kalman gain and the difference between the assigned location samples (perturbed) and the $\bar{x}^k$ calculated in the prediction step. This step is called state correction step and described by

$$\hat{x}^k = \bar{x}^k + K[z^k - H\bar{x}^k] \quad (10)$$

$$\hat{P}^k = \bar{P} - \bar{P}H^T(H\bar{P}H^T + S)^{-1}H\bar{P} \quad (11)$$

where $K = \hat{P}H^T S^{-1}$ is the Kalman gain. The so corrected state vector and covariance matrix are then fed back into the prediction equations for a new optimization problem at time k+1 and the steps are repeated for the next set of samples.

Let us define the following terms based on this problem description.

Perturbed Positions denotes the solutions $(\widetilde{x_n}(k), \widetilde{y_n}(k))$ for all users to the optimization problem. We refer to a series of them as *Perturbed Paths*.

Original Positions refers to $(x_n(k), y_n(k))$ for all users at time k and the series of them is named *Original Paths*.

The *Path Perturbation* Algorithm 1 returns perturbed paths from the original set of two users' paths. It maximizes instantaneous location privacy at each step by modifying the original set of location samples within the perturbation radius R. Larger R results in a higher degree of privacy, smaller R limits the effect of perturbation, which leads to higher quality of service and lower privacy.

We illustrate the use of the Path Perturbation algorithm in a simple scenario where two users travel on approximately parallel paths. If we randomly generate two paths for two users, we can divide the whole trip of two users into a finite number of samples.

Algorithm 1 $PathPerturbation$ calculates the set of perturbed location samples for two users, a 2 by B by 2 matrix, $PerturbedPaths$.

{Inputs: $OriginalPaths$, the set of continuous location samples for two users, a 2 by B by 2 matrix; R, perturbation circle radius as a user input; B, the segment size; process (user movement) and observation (tracking error) model for target tracking}
for $k = 1$ to B **do**
  for all hypothesis i **do**
    1. (State Prediction Step): Calculate the state prediction of each user based on parent tree.
  end for
  2. (Hypothesis Generation I): With the state prediction obtained in Step 1, set $equation(6)$ for every hypothesis i when perturbed paths were given.
  3. (Hypothesis Generation II): Set $equation(5)$ for every hypothesis i.
  4. (QoS Constraints): Set $equation(3)$ for every user n with R.
  5. (Solve the constrained optimization problem): Construct the cost function in $equation(4)$ with the result through Step 2 and 3. Set inequality constraints with the result of Step 4.
  for all ith hypothesis **do**
    6. (State Correction Step): Calculate the state update of each user based with perturbed samples.
    7. (Save Parent Probability): Obtained probabilities in Step 6 are saved for next probability tree.
  end for
end for

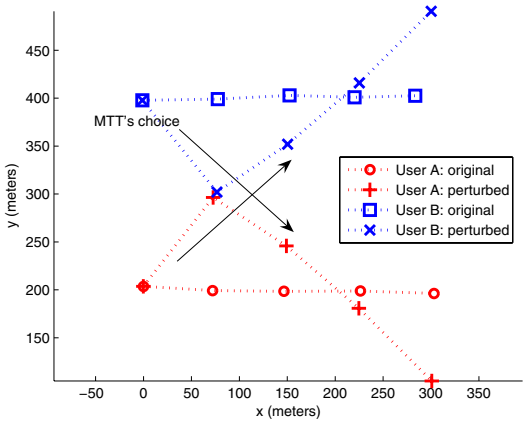

Figure 2. Two users move in parallel. The Path Perturbation algorithm perturbs the parallel segment into a crossing segment.

Figure 2 depicts the confusions that the Path Perturbation algorithm creates. The red circle points are periodic location samples from user A. The blue square points are those from user B. The crosses and X-marks are perturbed samples from user A and B respectively. Both users move from left to right starting out about 200 meters apart with a horizontal velocity of 15m/s. R is set to 100 meter. The algorithm assumes a correct assignment with probability 1 at the initial step. After that, the algorithm generates two hypothesis for each parent hypothesis, which was generated during the previous step. For the two user case, the algorithm must maintain 2^{k-1} hypothesis at step k. Starting at the second step, it tries to maximize the expectation of distance error, which leads to the conversion of parallel paths into crossing paths. The arrows in the figure show the result of applying the Multi Target Tracking algorithm to the perturbed data: the algorithm confuses the two users and follows the wrong track.

3.3 Path Segmentation

We apply the Path Perturbation algorithm only to selected segments of paths to allow the algorithm to scale to large numbers of users and longer paths. The number of hypotheses grows exponentially with path length, a scenario with N users and paths of length B yields $(N!)^B$ hypotheses. As a result, a Path Segmentation algorithm (see in Algorithm 2) must preprocess the data to identify short two-user segments, where the path perturbation algorithm can be applied.

In the following discussion, we refer to a segment as an area where the Path Perturbation algorithm could be applied. A segment is an area where two paths meet: a minimum of B consecutive location samples from two paths must lie near each other, with a maximum distance of $D = R\alpha$ between the samples of both paths (α is a scaling factor). A segment can be of either a crossing or a parallel type (in an approximate sense). We define a segment as crossing if the two paths intersect, and as parallel otherwise.

We must ensure that the path confusion approach cannot easily be inverted by an adversary. Assuming that the algorithms are known, the path confusion approach is only effective if the adversary cannot determine whether a path has been successfully perturbed. Otherwise, the adversary only needs to follow the *less* likely user to thwart this scheme.

The Path Perturbation algorithm performs best for short parallel segments. Therefore, we only apply it to such segments and leave naturally crossing segments unchanged. This can reduce the computation load but every crossing segment can be suspected of being an artificial crossing segment by an adversary. The characteristic of original traces (the frequency of occurrence of both segment types) can give a priori information to an adversary if one type is dominant over the other. Adversely, an adversary cannot distinguish artificial from natural ones if the frequency of occurrence of both segment types is comparable (see Figure 7).

The Path Segmentation algorithm proceeds as follows. At each step, N users report their location samples. The path segmentation algorithm keeps track of the distance between location samples at each step and then filters the $\frac{N(N-1)}{2}$ combinations into a candidate list that remained close enough for the last B steps. These candidate segments may contain segments with common users. Therefore, the list is filtered further until each user is part of at most one segment.[6]

PathSegmentation takes a matrix $In[2][K][N]$ as an input which is the set of original location samples of N users for K sample time. Adding to that, it takes in α (a scaling factor) and R from user. After segmentation, PathSegmentation outputs $Out[2][K][N]$ which is the set of perturbed location samples (two-dimensional) of N users. The PathSegmentation algorithm uses the following data structures as well as inputs and outputs variables:

$c[I][2]$: the set of combinations (2-subsets) out of N users, where $I = \frac{N(N-1)}{2}$

$d[I][K]$: the distance between the user positions of the ith combination at time index k

$flag[I]$: a boolean variable for indicating whether ith combination is a candidate segment

$b[I]$: the current size of temporary segment for ith combination

$segment1[2][B][I], segment2[2][B][I]$: two individual paths consisting of temporary segment for ith combination

The procedure CheckSegment estimates if the paths in the given segment cross each other. Its implementation approximates the two paths with lines through the start and endpoints of the path and tests whether the lines intersect. This assumes that paths do not change direction abruptly.

4 Evaluation

This evaluation studies the performance of the Path Perturbation algorithm using location traces from a random movement model. The algorithm must balance increased privacy protection against reduction in service quality that is caused by less accurate location samples. Therefore, the main evaluation metrics are mean location privacy and

[6] Among combinations has common users, pick the best combination to apply Path Perturbation algorithm and set $flag[i]$ unmarked for combinations who is not qualified (jth combination is better if $b[i] \leq b[i]$. If $b[i] == b[j]$, the combination who has smaller $d[i][k]$ is the best one.)

Algorithm 2 *PathSegmentation* returns perturbed set of N users' paths from the original set of N users' paths, taking advantage of subfunction, *PathPerturbationAlgorithm*.

{Input: $In[2][K][N]$, α, R, $process\ model$, $observation\ model$; Output: $Out[2][K][N]$; }
1. (Operating range of $Algorithm1$) $D \leftarrow \alpha * R$
2. (Initialization) Set all data structure to 0
3. (Compute the perturbed paths)
for $k = 1$ to K **do**
  // Find candidate segments based on distance
  for all $i, \forall i \in \{1, \ldots, I\}$ **do**
    Calculate every $d[i][k]$
    if $d[i][k] \leq D$ **then**
      $flag[i] \leftarrow 1$
    else
      $flag[i] \leftarrow 0$
    end if
  end for
  // Pick the best combination out of candidate segments to apply PathPerturbation
  $flag \leftarrow FilterCandidateSegments(flag, d, b)$
  // For not chosen combination,
  for all i such that $flag[i] == 0$ **do**
    $b[i] \leftarrow 0$
    // No operation is done on input
    $Out[\,][k][c[i][1]] \leftarrow In[\,][k][c[i][1]]$
    $Out[\,][k][c[i][2]] \leftarrow In[\,][k][c[i][2]]$
  end for
  // For chosen combination,
  for all i such that $flag[i] == 1$ **do**
    // Increase b[i] up to B
    $b[i] \leftarrow b[i] + 1$
    // Save the current location samples in temporary segments
    $segment1[\,][b[i]][c[i][1]] \leftarrow In[\,][k][c[i][1]]$
    $segment2[\,][b[i]][c[i][2]] \leftarrow In[\,][k][c[i][2]]$
    // If we gather B consecutive pairs of location samples, check if it is crossing or not and perturb it
    if $b[i] == B$ **then**
      $type \leftarrow CheckSegment(segment1, segment2, c[i][\,])$
      if $type == Crossing$ **then**
        // Leave it unchanged if crossing
        $Out[][k-b(i)+1:k][c[i][1]] \leftarrow path1[][1:b[i]][c[i][1]]$
        $Out[][k-b(i)+1:k][c[i][2]] \leftarrow path2[][1:b[i]][c[i][2]]$
      else
        // Perturb path segment
        $Out[\,][k - b(i) + 1 : k][c[i][1]] \leftarrow PathPerturbation(path1, path2, c[i][\,])$
        $Out[\,][k - b(i) + 1 : k][c[i][2]] \leftarrow PathPerturbation(path1, path2, c[i][\,])$
      end if
      $b[i] \leftarrow 0$
    else
      // We temporarily have output same to input but if we have B consecutive ones later, we overwrite the result of PathPerturbationw onto a matrix Out
      $Out[\,][k][c[i][1]] \leftarrow In[\,][k][c[i][1]]$
      $Out[\,][k][c[i][2]] \leftarrow In[\,][k][c[i][2]]$
    end if
  end for
end for

mean location error (quality-of-service), defined in equation (1) and in equation (2), respectively.

As a baseline for the evaluation we choose a gaussian random perturbation algorithm, which adds an iid gaussian offset to every location sample. We also show location privacy for unmodified location samples—labeled as "no operation" in the graphs. In scenarios with higher user density, location privacy will still increase even though these samples have not been processed by any privacy algorithm. This is because the available sampling frequency may not be high enough to allow a clear disambiguation of paths.

To obtain the location traces, we generate random scenarios using process and observation model parameters that reflect the relatively straight paths taken by vehicular traffic. The process model, which defines users' behavioral model, is represented by the following matrices, F and Q:

$$F = \begin{bmatrix} 1 & 0 & 5 & 0 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad Q = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix}$$

Matrix F causes the current horizontal and vertical position to be calculated with the previous position plus velocity, v multiplied by sampling interval $T_f = 5$. The state vector consists of x, y, v_x and v_y. We assume 5 meters of drift variance in position and 5m/sec of drift variance in velocity using process noise matrix Q. A larger process noise variance causes more changes in directions or velocity.

For creating the observation model which defines the representation of location samples and the accuracy of the underlying location tracking technology, we assume that every user reports only latitude and longitude of their current position. In addition, we assume that every user has a GPS receiver with the measurement error obeying $N(0, 5)$ in the car. The matrices, H and S formally express these assumptions:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad S = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}$$

We generate paths in a $1km^2$ area (i.e., one section of a road network) and assume that each car runs at 15m/s (54km/h). Since algorithm performance is dependent on user density we ensure that the number of users in this area remains constant during the experiment (i.e., that no users are leaving the square mile area). To this end, we randomly choose initial positions on the boundary of the area and randomly set initial trajectories within +/- 45 degrees towards the midpoint of the area. The number of users is set to $N = 5$, resulting in a user density of 12.8 per square mile. User density and sample frequency were chosen based on operational parameters found suitable for traffic monitoring applications by Cayford and colleagues [8]. Unless specified, the segment length B was set to 4. We generated 50

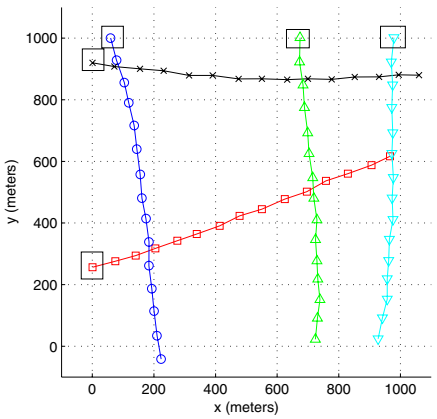

Figure 3. Example traffic $1km^2$ scenario for algorithm evaluation. Small rectangles indicate starting points of paths, which are randomly chosen on the boundary of the area.

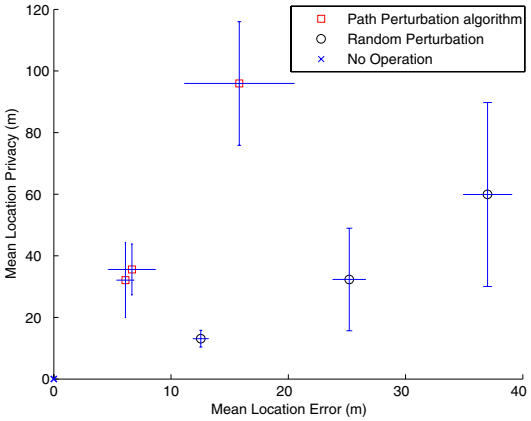

Figure 4. Mean Location privacy and mean location error for different perturbation constraints R (100m, 150m, 200m, from left to right). The Path Perturbation algorithm achieves higher quality of service for the same degree of privacy.

different random scenarios for our experiments. For illustration purposes, Fig. 3 shows one such example scenario.

4.1 Results

Figure 4 shows the mean location privacy versus the mean location error graph for three different schemes. Each point shows the mean computed from 50 randomly generated scenarios for a specific perturbation radius R setting. The radius settings are 100m, 150m, and 200m, from left to right, respectively. The error bars indicate a two standard deviations interval. This result shows that the Path Perturbation algorithm provides better location privacy than the random perturbation technique baseline for the same location error. From another perspective, if we compare two point with a similar location privacy level ($R = 150$), the Path Perturbation algorithm improve location error by about 20m. Consider the point with about 100m location privacy and about 16m location error. It means that 16m perturbation error for each user creates 100m location privacy for each user. Here, we show mean location privacy over the length of the path.

In Figure 5, however, we calculated the instantaneous location privacy at each sample time. The graph shows the mean results over all 50 scenarios. The result shows that privacy varies over time but tends to increase. For example, the path Perturbation algorithm achieves a maximum location privacy of 220m at any one point in the path that was represented by the point we considered in the previous figure. This value would likely increase in a longer simulation run, as the adversary is following the wrong path. In the depicted time interval, the perturbation algorithm is applied multiple times to the same path, which could explain the temporary reductions in privacy, as the adversary may stumble back on the correct path.

The Figure 6 illustrates the sensitivity of the results with respect to changes in block size B and perturbation radius R. These result were obtained through microbenchmarks comprising two short paths of 5 samples each with an approximate distance of $D = 200m$ between them. For simplicity privacy and QoS are shown as a single ratio value with higher values representing better performance. The smaller block size 3 leads to more variance in the outcome, while the results for block sizes of 4 and 5 are more stable across a broad range of different perturbation radius parameters.It clearly shows that parameters close to $\frac{D}{2}$, such as $R = 80m, 100m$ (i.e., $2 \leq \alpha = \frac{D}{R} \leq 2.5$), perform best. This experiment also indicates that performance may be further improved by adaptively varying the perturbation radius based on the distance between paths.

The following two figures present more data about the level of privacy afforded by our algorithm. These were obtained with an extended movement model, where the velocity of vehicles randomly varies between 20km/h and 60km/h. In Figure 7, crosses show the frequency of occurrence of parallel segments while square points depict the frequency of occurrence of parallel and crossing segments in our movement model. Each point is measured over 50 different scenarios and averaged out. Error bars indicate the standard deviation. The ratio of parallel segments to total

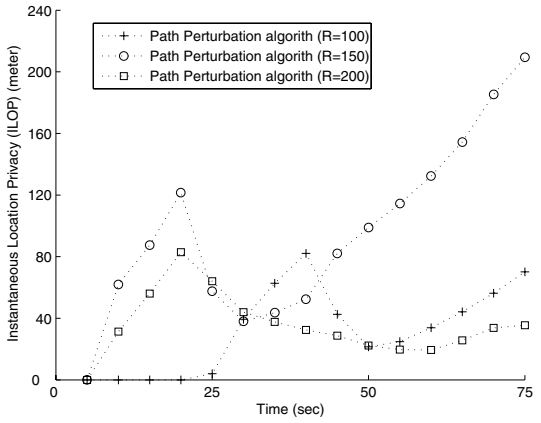

Figure 5. Instantaneous location privacy (ILOP) over time for three different values of R. Location privacy increases when the adversary follows the wrong track.

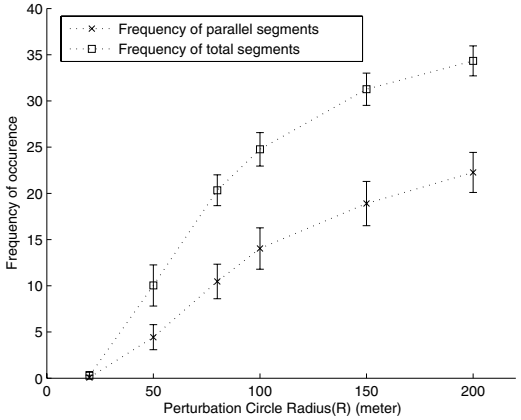

Figure 7. Frequency of occurrence of crossing and parallel segments given different perturbation radii R. Larger R produce more chances to apply the Path Perturbation algorithm. The frequency of parallel segments ranges from 45 percent to 65 percent to that of total segments.

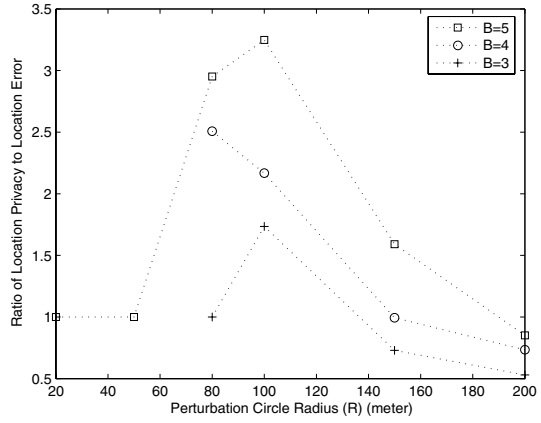

Figure 6. Sensitivity of algorithm performance to changes in block size B and perturbation radius R. Larger block sizes lead to more predictable privacy results.

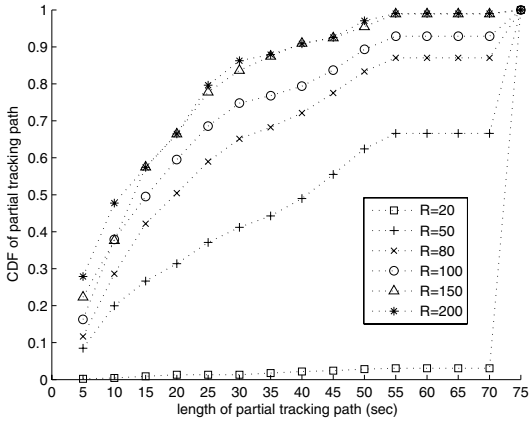

Figure 8. CDF of tracking time over different R. This graph shows how long an adversary can correctly track users' paths.

segments ranges from 45 percent to 65 percent over different R. A relatively balanced occurrence of both segments is important so that an adversary cannot distinguish artificially created crossings from real crossings through information on the prior distribution.

Figure 8 shows the cumulative distribution function for the length of time that an adversary could correctly track individual paths. Although the Path Perturbation algorithm keeps an adversary from learning the whole path of an individual user, even partial tracking can sometimes let an adversary infer private information. Figure 8 shows that 80 percent of partial tracking paths is less than 9 samples for $R \geq 80$, which corresponds to 45 seconds or 675m. Increasing R reduces this tracking time; for example, the 80 percentile is less than 25 seconds in the case of $R = 250$.

5 Discussion

The main result in figure 4 supports the hypothesis that path perturbation algorithms could increase privacy in traffic monitoring applications. The results were obtained at traffic densities that Cayford and colleagues [8] characterized as suitable for such a system. They further stated that these applications can tolerate up to 100m errors and still be able to distinguish road segments with 97.5% accuracy on local streets and 99% accuracy on freeways. If we assume that a GPS receiver produces location samples with 10m error a perturbation algorithm could introduce up to 90m errors, without significant detrimental effects. The Path Perturbation algorithm creates 100m location privacy at an average location error of only 16m, leaving a large margin of error for our simulation. In future work, this hypothesis should be tested with real location traces from vehicular traffic.

Although there exists a tradeoff between quality-of-service and location privacy, this study has shown that this relationship is not necessarily a zero-sum game. The statistical monitoring applications considered in this paper do not need to track individual users for extended periods of time. Thus, quality-of-service can be characterized as the error applied to each individual location sample. For privacy, user identity is important, therefore it must be defined over entire traces. A small perturbation on a few samples may lead an adversary onto the wrong track and lead to big improvements in privacy (if the tracks diverge) without the need to apply any further perturbation.

Given a quality-of-service constraint, path perturbation algorithms cannot eliminate the dependency on user density. Adequate levels of privacy can only be obtained if user density is sufficiently high. In a low user density environment, we either have to sacrifice QoS by increasing R or there may not be enough chances to apply the algorithm to obtain a suitable level of privacy. Figure 7 has shown that larger R create more chances for path confusion than smaller R given the same user density. Temporal perturbation provides another avenue for future investigations that allow operation in low user-density environments.

Privacy provided by the Path Perturbation algorithm also depends on the characteristics of the original traces, especially the frequency of parallel segments to crossing segments. If it is known that few crossing segments exist in the original paths, an adversary could assume that all crossing segments have been artificially inserted through a perturbation algorithm. In our random movement model this was not a problem (Figure 7) but this question should also be addressed through studies with real vehicular location traces. Privacy may further be compromised through advanced tracking algorithms that reject unlikely location samples. These algorithms would seek to identify perturbed samples and not consider them in the tracking equations. This could be addressed in the privacy algorithm through an additional per-step perturbation bound.

The algorithms used in this feasibility study are computationally too complex for a deployment in real-time information systems with large numbers of users. This overhead could be reduced by finding a closed-form solution to the optimization problem, for example through the Lagrange multiplier method. Another approach is to develop heuristic-based perturbation algorithms that approximate the performance of this solution. This would allow deployment of such algorithms in online applications.

6 Related Work

Prior work related to location privacy spans the fields of networking, pervasive computing, and data mining. Some of these mechanisms are not restricted to location privacy but generally applicable to other privacy problems. We focus our discussion on privacy mechanisms whose underlying theme is to degrade information in a controlled way before releasing it.

Beresford et al. [6] found that an adversary can often identify a user from anonymous path information by correlating it with knowledge about the environment. They proposed the mix zone concept in which a trusted proxy removes all samples before it passes location samples to application servers [7]. The degree of privacy offered by the mix zone was evaluated for pedestrian traffic under the assumption that an attacker uses empirical linking. However, the static mix zone concept cannot guarantee location privacy in the case that users' behavioral movement models have small variance (i.e., it is less probable that users change their direction in the mix zone) and in applications with low-user density. Our dynamic mechanisms can be adjusted for different privacy-quality-of-service tradeoffs. The perturbation approach can also provide a degree of un-

observability. This means that an adversary does not necessarily notice that privacy mechanisms are used, because no location samples are suppressed. Huang et al. [15] implemented a mix-zone concept by proposing the new concept of a silent period in which a station is not allowed to disclose its pseudonym.

Gruteser and Grundwald [13] reduce the spatio-temporal resolution of location-based queries to guarantee a defined degree of anonymity in different locations. These mechanisms assume that location-based queries are generated so infrequently, that they can be viewed as independent queries (the adversary should be unable to link them to the same user). The time-series nature of a continuous stream of location information poses novel privacy challenges that have not been addressed. Gruteser and Hoh [14] described how trajectory-based linking (i.e. Multi Target Tracking) can infer a user's path from individual location samples provided by several users. For database systems, Sweeney has argued that merely omitting obvious identifiers is not sufficient to ensure anonymity and has developed algorithms based on her k-anonymity concept [23, 24]

The same data degradation approach can be applied to the data mining privacy problem. Agrawal and Srikant [4] showed how a random perturbation technique can provide privacy for individual data items while still allowing reconstruction of the approximate distribution of values over a large number of users. Agrawal and Aggarawal [3] have provided an information theoretic metric to quantify the amount of privacy to take an adversaries prior knowledge into account. In this work, we showed that relying on adding white noise to location samples may fail because time series properties enable an MTT attack. Even for individual values Kargupta et al. [17] have pointed out that the spectral properties of white noise allow an approximate reconstruction of original data.

The question on how to define a privacy metric has led to another thread of privacy research. In the general data privacy area, Cynthia [9] suggested a framework for comparing privacy enhancing technologies in for preserving privacy in public databases. In the context of anonymous network communication, Serjantov and Danezis [22] as well as Diaz and colleagues [11] have proposed an information theoretic metric for anonymity. Anonymity is maximized when each subject is equally likely to have sent the message in question. This metric, however, is not readily applicable to continuous data such as location as opposed to discrete messages, because it does not take into account the difference between two data items.[7] Our work is based on a modified privacy metric that considers distance as well as assignment probabilities.

Prior work on location-aware systems has concentrated on privacy-policy mechanisms [19, 18, 12], where service providers publish policies that explain data collection and handling practices to potential users. Users can match privacy preferences with these policies. The IETF Geopriv group [10] is developing a standard for location privacy policy mechanisms. These efforts are complementary to our work. Privacy policies can establish trust between users and service providers, to ensure that collected data is not misused, service providers have to rely on security mechanisms such as access control or data anonymization. Privacy policies could also specify the degree of anonymity that service providers must maintain before they release information to third-party applications.

7 Conclusions

The prevalence of sophisticated location-tracking and wireless communication technology gives rise to a novel class of application that gather statistical information about people's movements. Current data perturbation techniques are unable to protect time-series location information. We have proposed a data perturbation technique that increases path confusion by slightly modifying reported positions for two users that are in close proximity of each other. This technique can limit the tracking duration, for which an adversary can follow an individual user. Specifically, we conclude that

- the Path Perturbation algorithm improves privacy with a lower mean location error, that means at a lower quality of service penalty than a gaussian perturbation algorithm.
- the Path Perturbation algorithms achieve promising results in an environment with about 10 vehicles per square mile, which is a user density targeted in traffic monitoring applications.

Future Work We see two promising avenues for further work. First, eliminating the reliance on a general optimization algorithm would improve computational efficiency of the Path Perturbation algorithm. We expect that heuristics can be found that retain most of the privacy and quality of service improvements but are much more efficient to compute. This would allow enable using the algorithms in applications with real-time data requirements. It would also improve scalability to scenarios with large number of moving users.

Second, applying the algorithms to datasets collected from real-life applications could provide an experimental validation of this approach. It will provide information

[7] Assume an adversary is given a set of location samples and a set of users. According to the entropy metric, users enjoy the highest degree of anonymity, if each user is equally likely to have generate each of the location samples. In terms of location privacy, this is meaningless if the location samples are too close each other, because the adversary knows all users locations without first assigning the samples to users.

about the relative frequency of crossing and parallel segments and can provide information on whether increased privacy can be achieved with acceptable data quality tradeoffs.

References

[1] Location privacy protection act. http://www.theorator.com/bills107/s1164.html, 2001.

[2] Wireless privacy protectection act. http://www.theorator.com/bills108/hr71.html, 2003.

[3] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Symposium on Principles of Database Systems*, 2001.

[4] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 439–450. ACM Press, May 2000.

[5] K. Ashok. *Estimation and Prediction of Time-Dependant Origin-Destination Flows*. PhD thesis, Massachusetts Institute of Technology.

[6] A. Beresford and F. Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, 2003.

[7] A. Beresford and F. Stajano. Mix zones: User privacy in location-aware services. In *IEEE Workshop on Pervasive Computing and Communication Security (PerSec)*, 2004.

[8] R. Cayford and T. Johnson. Operational parameters affecting use of anonymous cell phone tracking for generating traffic information. *Institute of transportation studies for the 82th TRB Annual Meeting*, 1(3):03–3865, Jan 2003.

[9] S. Chawla and C. Dwork. Toward privacy in public databases. In *The Second Theory of Cryptography Conference (to appear)*, 2005.

[10] J. Cuellar, J. Morris, and D. Mulligan. IETF geopriv requirements. http://www.ietf.org/html.charters/geopriv-charter.html, 2002.

[11] C. Diaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In *2nd Workshop on Privacy Enhancing Technologies*, 2002.

[12] S. Duri, M. Gruteser, X. Liu, P. Moskowitz, R. Perez, M. Singh, and J.-M. Tang. Framework for security and privacy in automotive telematics. In *2nd ACM International Worksphop on Mobile Commerce*, 2002.

[13] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the First International Conference on Mobile Systems, Applications, and Services*, 2003.

[14] M. Gruteser and B. Hoh. On the anonymity of periodic location samples. In *Proceedings of the Second International Conference on Security in Pervasive Computing*, 2005.

[15] L. Huang and H. Yamaneet. Enhancing wireless location privacy using silent period. In *5th Workshop on Privacy Enhancing Technologies*, 2005.

[16] S. Hughes. States mull taxing drivers by mile. http://www.cbsnews.com/stories/2005/02/14/eveningnews/main674120.shtml, Feb 2005.

[17] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. Random data perturbation techniques and privacy preserving data mining. In *IEEE International Conference on Data Mining*. IEEE Press, 2003.

[18] M. Langheinrich. A privacy awareness system for ubiquitous computing environments. In *4th International Conference on Ubiquitous Computing*, 2002.

[19] G. Myles, A. Friday, and N. Davies. Preserving privacy in environments with location-based applications. *IEEE Pervasive Computing*, 2(1):56–64, 2003.

[20] D. O. T. of Minnesota. Otso - about guidestars. http://www.dot.state.mn.us/guidestar/about.html, 1991.

[21] D. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, Dec 1979.

[22] A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In *2nd Workshop on Privacy Enhancing Technologies*, 2002.

[23] L. Sweeney. Achieving k-Anonymity Privacy Protection Using Generalization and Suppression. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):571–588, 2002.

[24] L. Sweeney. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[25] R. Vyas. Ford device intended to unclog roads. http://www.freep.com/money/autonews/ford27_20040227.htm, Feb 2004.