Locating Traffic Sensors on a Highway Network: Models and Algorithms

Fatemeh Sayyady, Yahya Fathi, George F. List, John R. Stone

Fatemeh Sayyady, Ph.D. (corresponding author)
Operations Research Program
North Carolina State University
484 Daniels Hall, Raleigh, NC 27695-7908
Phone: (919) 673-6088, Fax: (919) 515-5281
Email: fsayyad@ncsu.edu

Yahya Fathi, Ph.D.
Professor
Industrial and Systems Engineering Department
North Carolina State University
436 Daniels Hall, Raleigh, NC 27695-7908
Phone: (919) 515-6417, Fax: (919) 515-5281
Email: fathi@ncsu.edu

George F. List, Ph.D.
Professor
Institute for Transportation Research and Education
North Carolina State University
ITRE 3413, Building IV, Centennial Campus Box 8601
Raleigh, NC 27695-7908
Phone: (919) 515-8767, Fax: (919) 515-7908
Email: gflist@ncsu.edu

John R. Stone, Ph.D.
Professor
Department of Civil, Construction, and Environmental Engineering
North Carolina State University
422 Mann Hall, Raleigh, NC 27695-7908
Phone: (919) 515-7732, Fax: (919) 515-7908
Email: stone@ncsu.edu

Total Words (7744): Abstract (212) + Text (6032) + Figures (3) + Tables (3)
For the 92th Annual Meeting of the Transportation Research Board
First Submission: November 15, 2012
ABSTRACT

We consider the problem of finding optimal sensor locations on a traffic network so as to characterize system use overall. We study the problem under two practical scenarios. In the first scenario, we assume there is a given number of sensors (p) that we need to locate on the highway network. In this context, the problem is to find a collection of p locations among a given collection of candidate locations. In the second scenario, we assume that there is a cost (c_i) associated with installing a sensor at each candidate location i, and a total budget b. In this context, the problem is to find a collection of locations that provide the best possible characterization given the budget constraint. We propose a metric to evaluate a potential solution and then propose appropriate mathematical models for solving the problem for each scenario. We show that the budget-constrained problem is an extension of the well-known p-median problem. A new Lagrangian heuristic algorithm is presented to solve large instances of this problem where a budget constraint is imposed. Through a comprehensive computational experiment, we demonstrate that the Lagrangian heuristic algorithm provides solutions for large-scale networks within reasonable execution times. Examples are based on locating weigh-in-motion (WIM) sensors on a large-scale highway network.

INTRODUCTION

Despite today’s advances in instrumentation technology, it is still a challenge to find cost-effective ways to create observability for highway networks. Much of the US highway network is still not instrumented. For example, no one knows how many vehicles traverse most of the network segments each day let alone the number of trucks. Yet good information is key to effective and successful network management. The asset condition is driven by use, and resource allocation decisions are affected by relative use and deterioration rates, so having defensible and effective information about asset condition is critical for responsible fiscal management and refurbishment planning. Half of the data collection focuses on asset condition and rates of deterioration. The other half focuses on use and use patterns. This paper focuses on the latter, but both are critically important.

For most of the highway network, a spectrum of information about the traffic volume and loads, including vehicle counts, vehicle classifications, axle loads, and gross vehicle weight is needed to determine the use rate. In the context of monitoring traffic flows (volume), available literature breaks down into three categories: link flow estimation (1-3), origin-destination (OD) matrix estimation (4-9), and path flow estimation (10). While significant progress has been made on formulating and solving sensor location problems that estimate traffic flows, not much attention has been focused on monitoring traffic loads. Yet estimation of these axle loads is of great importance in developing prudent capital and operating budgets for the maintenance of pavements and bridges. It has been shown that non-truck vehicles have negligible effect on pavement deterioration and bridge damage (10). Therefore, we will focus on truck axle loadings for this research.

The focus of this paper is on locating sensors that are particularly effective in measuring axle load data (mainly truck axle load data). Among available surveillance systems, weight-in-motion (WIM) sensors are finding increasingly widespread use in highway network management due to their capability in collecting comprehensive range of traffic data including volume and loads data (11-18).

If WIM sensors were cheap and exhaustive deployment was easy, optimal placement of the sensors would be irrelevant. WIM sensors, however, require a controlled operating environment (such as a strong, smooth, and level pavement), and costly setup and calibration equipment. Hence, full instrumentation of the network is still cost prohibitive and technologically challenging. Under these conditions a reasonable alternative would be to install WIM sensors at a limited number of locations on the network and to infer from them the traffic load data (more specifically, the axle load data) for the entire network. Several questions arise in this context:

1. How do we determine the locations of the WIM sensors for monitoring traffic on the network?
2. How do we use the collected axle load data at the monitoring locations to infer the corresponding data for the remainder of the network?

3. How do we evaluate the effectiveness of the proposed method (inferring data from WIM sensors to other locations with no WIM sensor)?

In this paper we propose a strategy to answer these questions and to develop corresponding mathematical models and techniques for its implementation. We study the problem under two practical scenarios. In the first scenario, we assume there is a given number of sensors \( p \) that need to be located on the highway network. In this context, the problem is to find a collection of \( p \) locations among a given collection of candidate locations to place the sensors. In the second scenario, we assume that there is a cost \( (c_i) \) associated with installing a sensor at each candidate location \( i \), and a total budget \( b \). In this context, the problem is to find a collection of locations for the sensors such that the total installation cost does not exceed the allotted budget.

DATA COLLECTION, PARADIGM, AND METRICS

Truck Class Distribution

Before presenting the problem formulations, it is important to talk about how different sensors are able to record different type of data (volume and loads data). This will help the reader understand the purpose of the solution being sought.

Traffic agencies routinely operate relatively inexpensive sensors known as the Portable Traffic Counters (PTCs) throughout the traffic network to collect only the vehicle classification data, based on FHWA vehicle classification scheme that classifies vehicles into non-trucks (classes 1-3) and trucks (classes 4-13) \( (9) \). Technically any location on the network could be monitored using a PTC sensor, and traffic agencies typically select a relatively large number of such locations throughout the network based on their prior experience so as to obtain a comprehensive knowledge of the vehicle classification data on many road segments in the network. Throughout this article we assume that we have a relatively large set of locations on the network where we have already used PTC sensors and obtained their corresponding vehicle classification data. We refer to each such location on the network as a partially observed location, or a PTC location for short, and use \( N \) to represent the set of all such locations. We assume that there are \( n \) PTC locations on network (i.e., \( N = \{ PTC \ loc_1, PTC \ loc_2, PTC \ loc_3, ..., PTC \ loc_n \} \)).

A PTC sensor records the number of vehicles in each class based on the FHWA vehicle classification scheme. We define a truck class distribution for each PTC location on the network as a vector that specifies the percentage of each class of trucks at that location over a year. If the non-truck classes are omitted, the percentages for classes 4 to 13 need to be normalized so that they sum to 100\%. It follows that associated with each PTC location \( i \), there is a vector \( t_i = (t_{i,4}, t_{i,5}, ..., t_{i,13}) \) where \( t_{i,m} \) is the percentage of trucks in class \( m \) at the PTC location \( i \). Of course \( \sum_{m=4}^{13} t_{i,m} = 100 \).

To illustrate this notation we present a small numeric example using the data collected on the North Carolina (NC) highway network. In Table 1 we present the vehicle classification counts collected by PTC sensors at two arbitrary and distinct locations on the network. We also present the truck class distribution (normalized vectors) corresponding to these locations in Table 1.

We use these normalized vectors to compare the truck traffic patterns among various PTC locations on the network. More specifically, we use the Euclidean norm of the vector \( (t_i - t_j) \) as a measure of similarity of the truck traffic patterns between two locations \( i \) and \( j \), with respective truck class distribution vectors \( t_i \) and \( t_j \). We denote this Euclidean norm by \( d_{ij} \), and we have

\[
d_{ij} = \sqrt{\sum_{m=4}^{13} (t_{i,m} - t_{j,m})^2}.
\]

It follows that smaller values of \( d_{ij} \) indicate a higher degree of similarity between the truck class distribution vectors for two PTC locations \( i \) and \( j \), and larger values of this parameter imply the opposite.
FIGURE 1 FHWA Vehicle Classification Scheme (19)

TABLE 1 Vehicle Classification Counts and Truck Class Distributions at Two Arbitrary Locations in NC

<table>
<thead>
<tr>
<th>Vehicle Classification Counts</th>
<th>ID</th>
<th>Route</th>
<th>VC4</th>
<th>VC5</th>
<th>VC6</th>
<th>VC7</th>
<th>VC8</th>
<th>VC9</th>
<th>VC10</th>
<th>VC11</th>
<th>VC12</th>
<th>VC13</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>NC 64</td>
<td>23</td>
<td>146</td>
<td>35</td>
<td>1</td>
<td>72</td>
<td>175</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>462</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>I-40</td>
<td>516</td>
<td>1926</td>
<td>604</td>
<td>28</td>
<td>654</td>
<td>7766</td>
<td>86</td>
<td>385</td>
<td>178</td>
<td>39</td>
<td>12182</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Truck Class Distributions</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>NC 64</td>
<td>5</td>
<td>31</td>
<td>8</td>
<td>0</td>
<td>16</td>
<td>38</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>I-40</td>
<td>4</td>
<td>16</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>64</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>100%</td>
</tr>
</tbody>
</table>

VC: Vehicle Class

WIM sensors can collect comprehensive data about truck axle loads (or axle load distributions). Sayyady et al. (2011) showed that in the state of North Carolina locations with similar truck class distributions also have similar axle load distributions (20, 21). Ultimately, it is the axle load distributions that are of interest. But the observation by Sayyady et al. (2011) implies that the value of $d_{ij}$ as defined above can also be employed to measure the similarity of axle load patterns between any pair of locations $i$ and $j$ on the highway network in the state of North Carolina. We believe this assertion to be valid for the
traffic networks in other states as well, but we cannot be certain unless a similar analysis is carried out to ascertain its validity.

Throughout this paper we refer to \(d_{ij}\) as the truck pattern distance between locations \(i\) and \(j\). As stated above we assume that we have a given set \(N\) of locations on the traffic network and that the truck class distribution vector \(t_i\) for each location \(i \in N\) is known and given. Thus, we can easily determine the value of \(d_{ij}\) for every pair of locations \(i\) and \(j\) in the set \(N\).

### Allocation of PTC Locations

Our primary strategy for obtaining axle load distribution for the network is to install a limited number of WIM sensors on the network and to use the data obtained via these sensors to infer the axle load distribution for the entire network. To this end we limit our attention only to the set \(N\) of PTC locations on the network. Associated with each location (point) \(i\) in the set \(N\) we have a truck class distribution vector \(t_i\). We assume that \(N\) also represents the collection of locations on the network for which we wish to obtain axle load data. We further assume that a subset \(P\) of these locations is to be selected for installation of WIM sensors, i.e., \(P \subseteq N\), and that each PTC location \(i\) that does not have a WIM sensor is then assigned (allocated) to its nearest PTC location \(j\) that has a WIM sensor (i.e., in the set \(P\)) as measured by the truck pattern distance \(d_{ij}\). Clearly each location \(i\) in the set \(P\) is allocated to itself since the corresponding distance \(d_{ii} = 0\) for all \(i\). By assigning (or allocating) a PTC location \(i\) to its nearest PTC location \(j\) with a WIM sensor we imply that the axle load distribution for the two locations are sufficiently similar so that the collected axle load data at location \(j\) can be used for both locations. We now define a metric \(\delta(P)\) to measure the total truck pattern distance between the set of PTC locations in \(N\) and their corresponding locations in \(P\) as follows:

\[
\delta(P) = \sum_{i \in N} \min \{d_{ij}; j \in P\}
\]

We refer to this metric as the total similarity (or dissimilarity) associated with the subset \(P\). Smaller values of \(\delta(P)\) imply a high degree of similarity of truck class distribution vectors between the PTC locations and their corresponding (allocated) WIM sensors in the set \(P\). And by the previously quoted result of Sayyady et al. (2011) they also imply a high degree of similarity of their corresponding axle load distributions (20). In the remainder of this paper we use the metric \(\delta(P)\) to evaluate any subset \(P\) of the PTC locations \(N\) that we select for placement (installation) of WIM sensors in the context of obtaining the axle load distribution for the entire network. Our objective is to find a subset \(P\) of \(N\) with smallest value of \(\delta(P)\) within the restrictions that we discuss below.

### PROBLEM FORMULATION

We study the sensor location problem under two practical scenarios. We refer to the problems corresponding to these scenarios as the \(p\)-sensor location problem and \(b\)-sensor location problem.

#### The \(p\)-sensor Location Problem

In the context of first scenario, the problem is to find a collection of \(p\) locations among the given collection of candidate locations to place the WIM sensors. We refer to this problem as the \(p\)-sensor location problem. Given a set \(N\) of PTC locations and a distance \(d_{ij}\) between each pair of locations \(i\) and \(j\) in this set, the \(p\)-sensor location problem (or \(p\)-SLP) is to identify a collection \(P\) of cardinality \(p\) of these locations so as to minimize the corresponding \(\delta(P)\). This problem is structurally equivalent to a well-known problem in location theory known as the \(p\)-median problem (22). The problem can be formulated as a linear integer programming (LIP) problem, and efficient algorithms are already developed for solving
relatively large instances of this problem. Following is a description of this linear integer programming model.

**Integer Programming Model**

Let $x_{ij}$ and $y_j$ be 0–1 decision variables as defined below:

$$
y_j = \begin{cases} 
0 & \text{if a WIM sensor is installed at location } j \\
1 & \text{otherwise}
\end{cases}
$$

$$
x_{ij} = \begin{cases} 
0 & \text{if PTC location } i \text{ is assigned to location } j \text{ with WIM sensor} \\
1 & \text{otherwise}
\end{cases}
$$

The $p$-SLP can now be stated as the following linear integer programming (LIP) model that we refer to as model IP1:

$$\begin{align*}
\text{Minimize} & \quad \sum_{i \in N} \sum_{j \in N} d_{ij} x_{ij} \\
\text{Subject to} & \quad \sum_{j \in N} x_{ij} = 1 \quad \text{for all } i \in N \\
& \quad x_{ij} \leq y_j \quad \text{for all } i, j \in N \\
& \quad \sum_{j \in N} y_j = p \\
& \quad x_{ij}, y_j \in \{0,1\} \quad \text{for all } i, j \in N
\end{align*}$$

The objective is to minimize the total truck pattern distance between the PTC locations and their corresponding (assigned) WIM sensors. Constraint (1) states that each PTC location must be assigned to exactly one location with WIM sensor. Constraints (2) allow assignment only to locations where WIM sensors are installed. Constraint (3) states that the total number of WIM sensors should be equal to $p$. Constraint (4) sets the decision variables to be binary.

**Solution Methodology**

As stated earlier, this problem is known as the $p$-median problem in location theory. From the point of view of computational complexity, $p$-median problem belongs to the class of NP-hard problems (23). In practice, however, relatively small to medium size instances of the problem (e.g., with less than 500 PTC locations and with $p = 50$) can be solved by using existing commercial LIP solvers such as CPLEX 11. For larger instances of the problem this approach becomes ineffective due to its excessive CPU time and memory demands. For such larger instances several efficient algorithms have been developed. The reader is referred to the annotated bibliography of Reese (2006) for an extensive review of these algorithms (24).

We have solved relatively small to medium size instances of the $p$-sensor location problem, i.e., model IP1 (with less than 500 PTC locations) using CPLEX 11. For larger instances of the problem (with more than 500 PTC locations), we use the algorithm developed by Mulvey and Crowder (1979). Mulvey and Crowder relax the assignment constraint (Constrain 1) and employ an iterative subgradient procedure combined with a heuristic procedure to solve the problem (25). Mulvey’s algorithm is used because it is shown to perform efficiently for larger instances of the problem, and it is relatively easy to implement.

**The $b$-sensor Location Problem**

As described earlier the second scenario for the sensor location problem arises when a budget $b$ ($b > c_j$ for all $j$) is used to determine how many WIM sensors can be installed. We refer to the problem as the $b$-
sensor location problem (or $b$-SLP). In this problem our goal is to find a subset $P$ of the PTC locations with total installation cost less than or equal to $b$ so as to minimize $\delta(P)$.

This problem is similar to the $p$-SLP, but the presence of the budget constraint creates a critical difference for the corresponding solution algorithms, especially as it pertains to the design of effective algorithms for solving larger instances of the problem.

**Integer Programming Model**

The decision variables $x_{ij}$ and $y_j$ for this model are defined the same way as they were for the $p$-SLP. The $b$-SLP can be stated as the following LIP model that we refer to as model IP2.

$$
\text{Minimize } \sum_{i \in N} \sum_{j \in N} d_{ij} x_{ij} \\
\text{Subject to } \sum_{j \in N} x_{ij} = 1 \quad \text{for all } i \in N \\
x_{ij} \leq y_j \quad \text{for all } i, j \in N \\
\sum_{j \in N} c_j y_j \leq b \\
x_{ij}, y_j \in \{0,1\} \quad \text{for all } i, j \in N
$$

The difference between this model and model IP1 is that constraint (3) of IP1 has been replaced with constraint (7) where we limit the total installation cost to $b$.

From the point of view of computational complexity, the $b$-SLP is NP-hard because the $p$-median problem is a special case of this problem (where the installation costs are identical at all locations, i.e., $c_j = c$ for all $j \in N$), and that the $p$-median problem is known to be NP-hard. We were able to use CLPEX 11 to solve relatively small instances of the problem (with less than 300 PTC locations) on a PC with a 3 GHz Intel Pentium R processor, 1 GB RAM, and a 420 GB hard drive within reasonable execution time; but for larger instances of the problem we were not able to use this approach due to excessive memory demands. We had to develop a heuristic algorithm.

**$b$-SLP Solution Methodology**

In this section we propose a Lagrangian heuristic algorithm for solving this problem. The algorithm starts with a Lagrangian relaxation of model IP2 obtained by relaxing constraint (5). The optimal value of this Lagrangian relaxation provides a lower bound for the optimal value of model IP2. It then employs a subgradient search procedure to obtain the corresponding value of the Lagrange multipliers. Along with this procedure we also employ a primal heuristic procedure at every iteration to obtain a corresponding feasible solution for the problem (and thus a potentially smaller upper bound for the optimal value of its objective function). As the subgradient search continues, the gap between the upper bound and the lower bound obtained in this manner becomes smaller, and we terminate the algorithm either when this gap is smaller than a given threshold, or when a certain number of iterations are performed. Details of the algorithm follow.

**Lagrangian Relaxation.** First, Constraint (5) is relaxed to obtain the following model (referred to as LR):

$$
\text{Minimize } \sum_{i \in N} \sum_{j \in N} d_{ij} x_{ij} + \sum_{i \in N} \lambda_i (1 - \sum_{j \in N} x_{ij}) \quad \text{subject to (6)-(8)} \\
$$

where, $\lambda = (\lambda_1, ..., \lambda_n)$ is a vector of Lagrange multipliers. The terms of the objective function of model
LR can be rearranged, resulting in the following equivalent problem.

\[
L(\lambda) = \text{Minimize}_{x,y} \sum_{i \in N} \sum_{j \in N} (d_{ij} - \lambda_i) x_{ij} + \sum_{i \in N} \lambda_i \tag{LR}
\]

\[
x_{ij} \leq y_j \quad \text{for all } i, j \in N
\]

\[
\sum_{j \in N} c_j y_j \leq b
\]

\[
x_{ij}, y_j \in \{0,1\} \quad \text{for all } i, j \in N
\]

For any given vector \( \lambda = (\lambda_1, \ldots, \lambda_n) \) the following properties are true for all \( j \in N \) at every optimal solution of model LR: if \( y_j = 0 \) then \( x_{ij} = 0 \) for all \( i \in N \) and if \( y_j = 1 \) then \( x_{ij} = 1 \) for all \( i \) such that \( d_{ij} - \lambda_i < 0 \), and \( x_{ii} = 0 \) for all other \( i \). These properties allow us to replace all \( x_{ij} \) variables in model LR with \( y_j \) variables to obtain the following equivalent model:

\[
L(\lambda) = \text{Minimize}_{y} \sum_{j \in N} R_j y_j + \sum_{i \in N} \lambda_i
\]

\[
\sum_{j \in N} c_j y_j \leq b
\]

\[
y_j \in \{0,1\} \quad \text{for all } j \in N
\]

where, \( R_j(\lambda) = \sum_{i \in N} \min (d_{ij} - \lambda_i, 0) \).

Note that for any given vector \( \lambda \), the value of \( \sum_{i \in N} \lambda_i \) is just a constant, so it can be dropped from the objective function. Thereby the above model reduces to the following model, which is the well-known knapsack problem (KP).

\[
L'(\lambda) = \text{Minimize}_{y} \sum_{j \in N} R_j y_j, \text{ subject to } \sum_{j \in N} c_j y_j \leq b, y_j \in \{0,1\} \tag{KP}
\]

For a description of the knapsack problem and related algorithms see Martello and Toth (26). We obtain the optimal value of this knapsack problem, the optimal value of model LR is obtained as \( L(\lambda) = L'(\lambda) + \sum_{i \in N} \lambda_i \). Numerous efficient algorithms are developed to solve the knapsack problem (27-28). We solve this problem using a dynamic programing algorithm (28).

**Subgradient Search Procedure.** To obtain the largest lower bound via the Lagrangian relaxation we need to find a vector \( \lambda \) to

\[
\text{Maximize } L(\lambda) \text{ where } \lambda = (\lambda_1, \ldots, \lambda_n) \in \mathbb{R}^n \tag{LD}
\]

This problem is known as the Lagrangian dual of the original model IP2. We solve this problem using a subgradient search procedure. It is an iterative procedure that generates a sequence of Lagrange multiplier vectors \( \{\lambda^k\} \) that lead to the optimal solution of the Lagrangian dual problem (LD). This approach is similar to the approach used by Mulvey and Crowder (1979) in the context of the \( p \)-median problem. But the presence of the budget constraint in our model is a key difference in solving the corresponding sub-problems.

Given an initial vector \( \lambda^1 \), the subgradient procedure generates a sequence of Lagrange multiplier vectors \( \{\lambda^k\} \) by iteratively adjusting the Lagrange multipliers using the direction of subgradient vector:

\[
\lambda^{k+1} = \lambda^k + t_k g^k \tag{9}
\]
where $t_k$ is a positive scalar step size, and $g_k$ is a subgradient vector for $L(\lambda)$ at $\lambda^k$. We adopt the outline of this algorithm from Reeves (1993) and adjust its details to fit the situation here. The basic steps of the method are as follows:

**Initialization**

Initialize $Z \leftarrow -\infty$. Also, use a heuristic procedure to obtain a feasible solution for model IP2 and use its corresponding objective function value to initialize $Z$ (i.e., use a greedy algorithm to select as many PTC locations as possible within the limited budget $b$, then assign every PTC location to its nearest selected location). Let the iteration counter $k = 1$ for each $i$ and initialize the corresponding multipliers $\lambda^1_i$ arbitrarily, and set $\lambda^1 = (\lambda^1_1, ..., \lambda^1_n)$. Let $\pi$ be a user-defined parameter and set $\pi = 2$.

**Iterative step $k$**

1. Solve model LR with $\lambda = \lambda^k$ by solving the knapsack problem KP as described above to obtain its optimal objective value $L(\lambda^k)$. Let $y^k_j$ and $x^k_{ij}$ (for all $i, j \in N$) be the optimal solution of model LR.
2. If $L(\lambda^k) > Z$, let $Z \leftarrow L(\lambda^k)$.
3. Use the primal heuristic algorithm described below to obtain a feasible solution for model IP2 and its corresponding objective function value $Z'$. If $Z' < Z$, let $Z \leftarrow Z'$, otherwise keep $Z$ unchanged.
4. Determine the subgradient vector $g^k = (g^k_1, ..., g^k_n)$ using $g^k_i = 1 - \sum_{j \in N} x^k_{ij}$ for $i = 1, ..., n$.
5. Define the step size $t_k = \pi (1.05 \times Z - Z) / \sum_{i \in N} g^k_i$.
6. Let $\lambda^{k+1} = \lambda^k + t_k g^k$.
7. Let $k \leftarrow k + 1$ and repeat until the stopping criterion is satisfied.

Reeves (1993) suggests updating the value of parameter $\pi$ to half of its current value after every 30 iterations, and we observed that this updating procedure works well for this problem. We terminate the algorithm when the ratio of $(\overline{Z} - Z) / Z$ becomes smaller than $\varepsilon$ or when the number of iterations reaches a pre-specified number $I$.

**Primal Heuristic Algorithm.** This is a heuristic algorithm to obtain a feasible solution for the problem at each iteration of the subgradient procedure. At the beginning of each iteration in the above algorithm we obtain the optimal solution for model LR. Since constraint (5) is relaxed to obtain model LR, the resulting solution is not necessarily feasible for the original model IP2. In particular, the existing value of $x_{ij}$ might assign a PTC location $i$ to either “no WIM sensors” or to “several WIM sensors”. In this case we can obtain a corresponding feasible solution for model IP2 by assigning each PTC location to its nearest WIM sensor. If we use $S$ to denote the set of locations $j$ for which $y_j = 1$ (i.e., the set of locations with WIM sensors), for each $i$ let $x_i^* = 1$ for $j^* = \arg\min_{j \in S} (d_{ij})$ - break ties arbitrarily, and let $x_{ij} = 0$ for all other $j$. The resulting solution is now feasible for the original model IP2, and the corresponding objective function value is thereby an upper bound for the optimal value of model IP2.

**COMPUTATIONAL EXPERIMENT**

A set of computational experiments was conducted to evaluate the effectiveness of the new Lagrangian heuristic algorithm. The hypothetical problems involved relatively large instances of the b-SLP (i.e., instances for which the number of PTC locations is greater than 300).
The Hypothetical Instances

We constructed two collections of randomly generated hypothetical instances of the problem that we refer to as unstructured and structured instances. For the first set, the truck class distribution vectors do not follow any specific pattern and all the elements of these vectors are uniformly distributed in a given range. For the second collection of instances, a natural clustering is imposed to provide illustrations more consistent with observed truck traffic patterns on the highway networks (29).

Unstructured Instances

Each instance of the b-SLP is identified by a given set of n PTC locations, a vector of truck class distribution vector $t_i$ and an installation cost $c_i$ associated with each location $i$ (for $i = 1, ..., n$), and a given budget $b$. For each location $i$, we randomly generate 10 distinct integer values representing the vehicle classification counts of the 10 truck classes by using the Uniform distributions presented in Table 2. These 10 integer values are then normalized to obtain the corresponding truck class distribution vector at that location.

We also determined the cost of installing a WIM sensor at each location $i$ (i.e., $c_i$) by using a uniformly distributed random integer between 30 and 40 (presumed to be in units of $1000). The reason for choosing this interval was that the average installation cost of a WIM sensor in a 4-lane roadway was known to be $35,000 (30), and that according to our best estimates, the actual cost at each location did not deviate from the average value by more than $5,000.

TABLE 2 Distributions used to Develop the Truck Use Vectors for the Unstructured/Structured Instances

<table>
<thead>
<tr>
<th>Vehicle Class</th>
<th>Unstructured</th>
<th>Structured</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set 1</td>
<td>Set 2</td>
</tr>
<tr>
<td>VC4, VC7, VC12, VC13</td>
<td>Uniform(5,50)</td>
<td>Uniform(5,50)</td>
</tr>
<tr>
<td>VC6, VC8, VC10, VC11</td>
<td>Uniform(5,100)</td>
<td>Uniform(5,100)</td>
</tr>
<tr>
<td>VC5</td>
<td>Uniform(100,1000)</td>
<td>Uniform(300,700)</td>
</tr>
<tr>
<td>VC9</td>
<td>Uniform(100,1000)</td>
<td>Uniform(300,700)</td>
</tr>
</tbody>
</table>

VC: Vehicle Class
Uniform($a,b$); $a$ and $b$ are boundaries of the uniform distribution.

Structured Instances

The structured instances were based on Sayyady et al. (2011). In North Carolina, the truck class distributions are classified into three distinct clusters as shown in Figure 2 (29). In cluster 1, the percentage of vehicles in class 5 (short-haul trucks) and class 9 (long-haul trucks) are fairly similar. Roadways with such truck class distributions are mostly located in the central region of NC. In cluster 2, the percentage of class 9 vehicles is appreciably higher than that of class 5 vehicle. Such pattern is common on Interstates and US highways that serve mostly the long-haul trips. In cluster 3, the percentage of class 5 vehicles is higher than that of class 9 vehicles. Such a traffic pattern is common on rural-recreational roads located in the mountainous region of NC.

Each structured instance was created by first dividing the locations into three groups each comprising roughly $n/3$ locations. Each group was meant to represent one of the three distinct traffic clusters in NC. For each PTC location in groups 1, 2, and 3 we used the Uniform distributions of sets 1, 2, and 3 (presented in Table 2), respectively, to generate the corresponding truck class distribution at that location.

In addition, for each location, we determined the cost of installing a WIM sensors as described above.
The Experiments

We construct a total of 30 randomly generated instances of the $b$-SLP of varying sizes. More specifically, we construct two collections of truck class distribution vectors (one according to the unstructured pattern and one according to the structured pattern) for each of five values of $n = 100, 300, 500, 700, \text{ and } 1000$. For each of the resulting ten collections of vectors we construct 3 instances of the problem with $b = 70, 350, \text{ and } 1750$, respectively.

We solved the smaller instances (with $n \leq 300$) using CPLEX 11 as well as our new Lagrangian heuristic algorithm. We evaluated the quality of the solution obtained via the Lagrangian heuristic by the value of $\rho = (\bar{Z} - Z_{opt})/Z_{opt}$, where $Z_{opt}$ is the optimal objective value of model IP2 obtained using CPLEX 11. We refer to $\rho$ as the performance ratio. For the larger instances (i.e., $n > 300$) we were not able to solve model IP2 via CPLEX 11 due to excessive memory demands. Hence we solved these instances only via the Lagrangian heuristic. For these instances we evaluated the quality of the solutions by the value of $\rho' = (\bar{Z} - Z)/Z$. Obviously $\rho \leq \rho'$ for each instance. For the Lagrangian heuristic algorithm we terminate the algorithm when the value of parameter $\rho' \leq 0.01$ or when the number of iterations reaches 1000, whichever occurred first.

The Results

Results of the computational experiment are summarized in Table 3. We make the following observations based on the results:

- The performance of the Lagrangian heuristic algorithm is reasonably good. The value of parameter $\rho$
(or \(\rho\)) is less than 0.010 for most problem instances (24 instances out of 30), i.e., the algorithm often
locates solutions that are very close to the optimal solution. Occasionally it even locates the optimal
solution (i.e., \(\rho = 0\)). The worst reported performance ratio is \(\rho' = 0.020\) (for the problem
instance with \(n = 500\) and \(b = 350\)).

- The structure of the data set (i.e., structured or unstructured) does not seem to have an appreciable
  impact on the performance of the algorithm.
- The execution time for the Lagrangian heuristic algorithm is relatively small; for most instances in
  our experiment it is only a few seconds, and the largest execution time observed is less than six
  minutes. Naturally, the execution time tends to increase as we increase the size of the instances (\(n\)).
  The execution time also seems to increase as we increase the value of \(b\) (for a fixed \(n\)).
- The execution time of the Lagrangian heuristic algorithm is smaller than that of CPLEX 11 when a
direct comparison can be made (i.e., for the smaller instances). For larger instances the corresponding
execution time of the Lagrangian heuristic algorithm remains relatively low (344 seconds for the
largest instance). To some extent the magnitude of the execution time for this heuristic algorithm
depends on the stopping rules employed. Intuitively, a higher threshold value \(\mathcal{E}\) or a smaller limit on
the number of iterations (\(I\)) would result in a lower execution time.

We also obtained the average execution time of the Lagrangian heuristic algorithm for all instances in
each size \(n\), for different values of \(n\). We observed that the average execution time increases as a function
of \(n\), seemingly at a modest increasing rate. A key component of the execution time for this algorithm is
the time we spend in solving the knapsack problem for each iteration. Replacing the dynamic
programming algorithm with an appropriate heuristic algorithm could have an appreciable impact on
reducing its execution time, albeit at the cost of a possible deterioration in the quality of the resulting
solutions.

<table>
<thead>
<tr>
<th>(n)</th>
<th>(b)</th>
<th>(\rho \text{ or } \rho')</th>
<th>Unstructured Instances</th>
<th>Structured Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Iterations</td>
<td>Time-L</td>
<td>Time-C</td>
</tr>
<tr>
<td>100</td>
<td>70</td>
<td>0.001</td>
<td>100</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>350</td>
<td>0.008</td>
<td>244</td>
<td>2</td>
</tr>
<tr>
<td>100</td>
<td>1750</td>
<td>0.009</td>
<td>90</td>
<td>3</td>
</tr>
<tr>
<td>300</td>
<td>70</td>
<td>0.000</td>
<td>178</td>
<td>3</td>
</tr>
<tr>
<td>300</td>
<td>350</td>
<td>0.009</td>
<td>350</td>
<td>8</td>
</tr>
<tr>
<td>300</td>
<td>1750</td>
<td>0.009</td>
<td>200</td>
<td>17</td>
</tr>
<tr>
<td>500</td>
<td>70</td>
<td>0.001</td>
<td>102</td>
<td>3</td>
</tr>
<tr>
<td>500</td>
<td>350</td>
<td>0.020</td>
<td>1000</td>
<td>47</td>
</tr>
<tr>
<td>500</td>
<td>1750</td>
<td>0.008</td>
<td>221</td>
<td>32</td>
</tr>
<tr>
<td>700</td>
<td>70</td>
<td>0.000</td>
<td>93</td>
<td>5</td>
</tr>
<tr>
<td>700</td>
<td>350</td>
<td>0.019</td>
<td>1000</td>
<td>57</td>
</tr>
<tr>
<td>700</td>
<td>1750</td>
<td>0.009</td>
<td>243</td>
<td>53</td>
</tr>
<tr>
<td>1000</td>
<td>70</td>
<td>0.000</td>
<td>114</td>
<td>12</td>
</tr>
<tr>
<td>1000</td>
<td>350</td>
<td>0.005</td>
<td>253</td>
<td>34</td>
</tr>
<tr>
<td>1000</td>
<td>1750</td>
<td>0.013</td>
<td>1000</td>
<td>344</td>
</tr>
</tbody>
</table>

Time-L: The total execution time (in seconds) that the Lagrangian heuristic algorithm takes to solve an instance of the problem.
Time-C: The total execution time (in seconds) that CPLEX 11 takes to solve an instance of the problem.
\(\rho\) or \(\rho'\): The performance ratio; for the instances with \(n \leq 300\) we report the value of parameter \(\rho\), and for the instances with \(n > 300\) we report the value of parameter \(\rho'\).
It is also true that the stopping criterion has an impact on the execution time of the algorithm as well as on the quality of the resulting solution. To investigate, we let the algorithm terminate only when the number of iterations reaches 1000 and we record the value of parameter $\rho$ (or $\rho'$). In some instances, we observed a further decrease in the values of $\rho$ or $\rho'$. For example, for one instance with $n = 500$ and $b = 1750$, the value of $\rho'$ decreases from 0.008 to 0.004. In some instances the sequence of upper bounds becomes closer to the corresponding lower bounds if we allow the algorithm to run for a larger number of iterations. This observation is of particular importance when the execution time is not a limiting factor.

REAL-WORLD EXAMPLE

The Traffic Survey Unit (TSU) at North Carolina Department of Transportation (NCDOT) is required to install WIM sensors on various roads in the State of NC highway network as part of their traffic monitoring activities (31). Their challenge is to decide on the collection of locations to install WIM sensors such that the total installation cost does not exceed a budget amount. To support TSU in their decision making process, we constructed an instance of $b$-SLP using NC traffic data. This instance consisted of 644 PTC locations in the North Carolina highway network (i.e., $n = 644$).

We obtained the truck class distribution at every PTC location using the vehicle classification counts collected at these locations. For each PTC location, we also determined the associated cost of installing a WIM sensor at that location as a randomly generated integer between 30 and 40 as previously described. We choose $b = 1000$ (in units of $1000$). The data associated with this instance is available from the authors upon request.

We used the Lagrangian heuristic algorithm presented earlier to solve this problem. The algorithm terminated after 368 iterations. For this number of iterations, we observed a solution for which the value of parameter $\rho' = 0.007$. The execution time for this problem was 84 seconds. The solution identified 28 locations where WIM sensors should be installed (given the budget of $1,000,000$).

For this specific instance, we also opted for a different stopping criteria (i.e. we set the algorithm to terminate after one hour) and recorded the value of $\rho'$. We observed that the value of parameter $\rho'$ further decreased from 0.007 to 0.006 after 1600 iterations and it remained the same thereafter. Since the execution time is not a limiting factor in such real-world instances, if we let the algorithm run for some allowable time frame it may find a more favorable solution.
CONCLUSIONS AND RECOMMENDATIONS

In this paper, we described a strategy for solving the sensor location problem under two practical scenarios. In the first scenario, there is a given number of sensors \( p \) that need to be located on the highway network. In the second scenario, there is a cost \( c_i \) associated with installing a sensor at each candidate location \( i \), and a total budget \( b \). We developed all corresponding mathematical models for implementing this strategy in each scenario and suggested effective algorithms to solve these models. The Lagrangian heuristic algorithm performs effectively for all instances in the second scenario. The Lagrangian heuristic often identifies a solution that is very close to optimal for small instances of the problem (the performance ratio is less than 0.010). Occasionally, it even provides evidence that the solution it obtains is optimal. The Lagrangian heuristic algorithm also performs effectively for solving instances of real-world size.

Our sense is that the new heuristic has significant merit, can be applied fairly easily, and requires only modest information about the traffic on the highway network. We recommend traffic agencies try to implement our proposed strategy in a phased way and gradually incorporate it into their current practices (for selecting WIM locations). Perhaps, a phased implementation from a limited set of high priority candidate locations to a larger set of locations could be considered as the agencies become more familiar with the method and gain experience with the process. We further recommend that the highway agencies outside of the state of North Carolina carry out an analysis similar to that in Sayyady et. al. (2011) to ascertain the validity of the resulting observations on their corresponding highway networks.

ACKNOWLEDGMENTS

The authors are grateful to Kent Taylor the head of TSU at the NCDOT for his motivation and creative vision. Particular thanks are due to the staff at TSU who collected and processed the data.
REFERENCES


TRB 2013 Annual Meeting Paper revised from original submittal.
19. Federal Highway Administration, FHWA Vehicle Types


