Wireless Sensor Network Optimization in Wildfire Detection

Jianxing Liao, Chong Wang and Xinyu Zhang
Washington University in St. Louis, liaojianxing, chong.wang, xinyuzhang@wustl.edu

***Abstract* – Wireless Sensor Network often faces challenges in the real-world application due to limited battery power. From many previous works, it is shown that the probabilistic model based on the coverage maintenance protocol is able to prolong the lifetime of the network. In this paper, our discussion extends from such a probabilistic model. We define an approximate solution to the sensor nodes deployment and also modify the original model so that the novel model is capable of monitoring and predicting forest wildfire, which has been a serious issue for many years in the Western United States. Our work heavily emphasizes optimization of the performance of wireless sensor networks. The solution quality has been improved by incorporating an additional probability distribution of the wildfire occurrence, which is obtained by the model learned from logistic regression. The computation time to obtain a solution in a reconfiguration phase has also been analyzed, and three different approaches have been developed, aiming to increase the computation speed. Among all those approaches we developed, the Jensen's inequality provides an approximate solution but has the lowest computation cost. We design two experiments to validate the performance of our optimization. The experiments' result indicates that our work has great potential when applying to real-world problems.**

Index Terms – Gradient Descent, Jensen's Inequality, Logistic Regression, Wireless Sensor Network

INTRODUCTION

The recent wildfire in California has caused many deaths and has raised concerns about the safety of people. Wildfire is a common and frequent problem in California so that it is important to take best actions possible to monitor the forest and prevent forest fire. In order to successfully monitor the wildfire issue in the forest, a wireless sensor network can be deployed in the forest. Such a network consists of sensor nodes, with each equipped with a sensing measurement tool, communication model, and power unit. However, equipped with the battery power, each sensor node will only be able to work within a limited time. Therefore, many previous works have been done to prolong the lifetime of the wireless sensor network.

Among all those efforts, the coverage maintenance protocol is proved to be capable of effectively increasing the wireless sensor network operation time. The key idea of this protocol is to only activate a certain number of nodes to monitor while scheduling the other nodes to sleep to conserve the energy. Furthermore, when the current network configuration fails to cover the region it is monitoring, a quick reconfiguration is required. The research article "Efficient Coverage Maintenance Based on Probabilistic Distributed Detection" [1] follows the idea of coverage maintenance protocol and gives a promising solution to reconfigure the network. And in our paper, we extend the idea presented in this research, modifying the model so that the network could be used to monitor the wildfire, introducing the logistic regression to improve the solution quality, and developing algorithms to lower the computation cost.

PREVIOUS WORKS

Typically, all the models which follow the coverage maintenance protocol fall into two categories — deterministic model and probabilistic model. As its name stands, the first one predicts a presence of a target in the network based on a perfect sensing assumption. Each node in such a network has a perfect sensing range. And any target showing within the sensing range is guaranteed to be detected and reported by the network. The limitation of the deterministic model is obvious. In the real world, a network is supposed to be deployed in an environment where noise is normally expected. Moreover, we also should admit the fact that no sensor equipment works perfectly anytime and anywhere. Thus, the system may and, indeed, should be allowed to make mistakes, either for identifying a nonexistent target or missing a true target.

Based on the above discussion, a probabilistic model will be more reasonable to measure the performance of the sensor network. In this paper, we modify probabilistic model in the research article "Efficient Coverage Maintenance Based on Probabilistic Distributed Detection" [1] so that it fits into the forest wildfire case with a better performance.

Firstly, we re-emphasize the two most important probabilities, the system overall detection probability, P_D, and false alarm rate, P_F. Given a target, which is wildfire in our case, at location (x, y), the wireless sensor network will have a probability $P_D(x, y)$ to actually detect this target. And if this probability is higher than a given threshold β for any location in the specified region A, then we will announce that the current network configuration is sufficient to detect any target occurring in this network. Also, as mentioned, the noise and all the other unexpected factors in the real world may negatively affect the decision of the network and, thus, results in a false judgment. Therefore, the probability of making a

wrong decision, P_F, has to be lower than a given threshold α. Thus, if our wireless sensor network is able to cover the whole geographic region, specifically, to be able to detect any fire in the forest, then the following expression has to be satisfied:

$$(\forall(x,y) \in A,\ P_D(x,y) \geq \beta) \wedge (P_F \leq \alpha) \quad (1)$$

The system false alarm rate is given based on the problem itself. As the forest campfire mostly leads to serious damage, the one should not give a very large system false alarm rate. For instance, suppose a fire occurred, then the probability that the system missed this target should be very low. Therefore, a reasonable low P_F would be more desirable. The specific value should be based on the empirical experiment, thus, will not be discussed in this paper.

The number of nodes should also have already been determined and set up based on each specific situation, such as the range that the network is going to cover or the topological complexity. It is worth to note that the approximate location of where to set up each node is very important because the location of each node strongly affects the result in the reconfiguration phase. The placement of sensors is presented under the section below.

PLACEMENT OF SENSORS

First of all, the assumption is made that each sensor can cover places that are within r meters from the sensor. In other words, the coverage area of each sensor is a circle that has a radius of r meters. In order to make the sensors to cover the entire region and to deal with emergency situation when one of the sensors runs out of battery, it is important to make sure that every point in the grid can be covered by at least two sensors. This problem can be referred to as the k-coverage problem (2-coverage in this case), which aims to minimize the total number of sensors in a given region following the condition that each point in the grid is covered by at least k sensors. The k-coverage problem, however, is a NP-hard problem as discussed in the paper "Randomized k-Coverage Algorithms for Dense Sensor Networks" [2]. The k-coverage problem takes significant computational effort and our following work will still minimize the number of active sensors in the network, so it is assumed that all the sensors are initially evenly spread out in the grid, as suggested in the figure below. If each sensor can cover an area of radius r meters, then the sensors are evenly placed at r meters apart from each other.

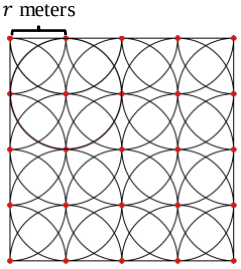


FIGURE I
SENSOR PLACEMENT DEMONSTRATION

In the diagram, it is obvious that all the places in the grid are covered by at least two sensors, which are represented by red dots in Figure I, guaranteeing that if one of the sensors runs out of battery, each point can still be covered by at least one sensor. In case of such kind of emergency, our model will quickly recalculate and reconfigure the entire sensor network to make sure that the algorithm still selects the minimum number of sensors needed to cover the entire region.

PROBABILITY MODEL

I. Modification from Previous Model

In a reconfiguration phase, the challenge is to schedule as few as many numbers of nodes to monitor while keeping the rest nodes to sleep in order to conserve the energy. And meanwhile, the whole system still has to achieve a certain level of performance. The specific algorithm is designed and improved later. And we first introduce the probability model itself in this section.

The assumption made is that the location of a target is unknown, but each node has the ability to identify its own location. The sensor equipped on the node obviously differs based on the environment the sensor network is monitoring. Here, as we are monitoring the campfire occurring in the forest, the power of a target at location (x, y) would be represented by the radiation intensity with respect to the distance. As discussed in the paper "Temperature Field Modeling and Simulation of Wireless Sensor Network Behavior during a Spreading Wildfire" [3], the equation is given by the expression below, where the $e(x_i, y_i)$ is the energy detected at (x_i, y_i) and was emitted at (x, y):

$$e(x_i, y_i) = 60 \times \left(1 - e^{-\frac{L^2}{d((x,y),(x_i,y_i))}}\right) \quad (2)$$

where L is the flame length.

As suggested in previous sections, to deploy such a wireless sensor network in the real world, it should be able to tolerate the effects caused by unexpected factors, such as noise. In our case, one can imagine any heat resources in the forest other than the fire as the noise, such as human beings' and animals' activities. The sensor itself should also not be expected to perform perfectly all the time. But all these noises are hard to model and beyond the scope of this paper. Therefore, we simply model all the noises as an overall noise effect, which is coincident with the assumption made in the "Efficient Coverage Maintenance Based on Probabilistic Distributed Detection" [1]. The hypothesizes that a target is absent and present are as following:

$$H^0: p(z_i|H^0) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{z_i^2}{2\sigma^2}} \quad (3)$$

$$H^1: p(z_i|H^1) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\left(z_i - \sqrt{e(x_i,y_i)}\right)^2}{2\sigma^2}} \quad (4)$$

where z_i is the measurement made by node i, and when there is no actual signal, the noise is modeled as the Gaussian distribution, with mean of 0.

Before computing the overall detection probability and system false alarm rate, each node has to compute its own detection probability and local system false alarm rate. The majority rule will be used later to determine the overall probabilities. The original work gives

$$P_{F_i} = \int_{\lambda_i^*}^{\infty} p(z_i|H^0)\, dz_i = Q\left(\frac{\lambda_i^*}{\sigma}\right) \quad (5)$$

$$P_{D_i}(x,y) = \int_{\lambda_i^*}^{\infty} p(z_i|H^1)\, dz_i = Q\left(Q^{-1}(P_{F_i}) - \frac{\sqrt{e(x_i,y_i)}}{\sigma}\right) \quad (6)$$

The equations assume a uniform probability distribution of having a target for each location. However, in reality, some locations obviously are more likely to have wildfire, depending on the vegetarian types, human beings' activities frequency, date, weather and so on. Thus, to accurately model probability of having a target at a specific location, an additional probability distribution needs to be considered. In our paper, we use a historical data set to learn a logistic regression model, which will be applied to determine the possibility of having a wildfire on a location based on its own features. The specific procedure to train the model is specified under the sub-section IV.

We notice that the decision threshold is equal to the ratio of prior probabilities of no fire versus fire, where the prior probability can be modelled by our logistic regression model. We can use this result to directly compute the decision threshold without solving for the limit of a Gaussian CDF, as was done in the previous paper [1].

This algorithm uses logistic regression to model the likelihood ratio. To perform Maximum A Posteriori reasoning, we must have a node activate when satisfying these inequalities:

$$P(H_1|z_i) \geq P(H_0|z_i) \quad (7)$$

$$P(H_1)P(z_i|H_1) \geq P(H_0)P(z_i|H_0) \quad (8)$$

$$\frac{P(z_i|H_1)}{P(z_i|H_0)} \geq \frac{P(H_0)}{P(H_1)} = \frac{1-P\left(y=1|\theta_{logistic}\right)}{P\left(y=1|\theta_{logistic}\right)} = \lambda_i^* \quad (9)$$

II. Unmodified Part of Previous Model

After all the modifications in the above section, the procedure to compute the P_D for each location and the overall $P_{D_{min}}$ would be identical to the work in "Efficient Coverage Maintenance Based on Probabilistic Distributed Detection" [1]. Since our paper aims to compare the performance of sensor network with and without the distribution obtained based on the logistic regression model and the overall computation time, we simply list formulas required for the computation below. Readers may refer the original paper for further clarification.

$$\sum_{|S_1|>|S_0|} \prod_{i\in S_0}(1-\alpha_0) \prod_{j\in S_1} \alpha_0 \leq \alpha \quad (10)$$

$$P_D(x,y) = \sum_{|S_1|>|S_0|} \prod_{i\in S_0} \left(1 - P_{D_i}(x,y)\right) \prod_{j\in S_1} P_{D_j}(x,y) \quad (11)$$

$$P_F = \sum_{|S_1|>|S_0|} \prod_{i\in S_0}\left(1 - P_{F_i}\right) \prod_{j\in S_1} P_{F_j} \quad (12)$$

$$P_{D_{min}} = \min_{(x,y)\in A} \sum_{|S_1|>|S_0|} \prod_{i\in S_0} \left(1 - P_{D_i}(x,y)\right) \prod_{j\in S_1} P_{D_j}(x,y) \quad (13)$$

III. Minimum Detection Probability Algorithm

The pseudo code for the algorithm introduced in the previous work, combined with our modification is expressed as follows:

PSEUDO CODE I

MINIMUM DETECTION PROBABILITY ALGORITHM

begin
Given total number of active nodes, system false alarm rate, and geographic region A
Initialize an empty 2D array of size being equal to the total number of locations in region A
for each node i in active nodes
for each location (x, y) in region A
Local false alarm rate <— compute equation (5)
Assume each node has the same local false alarm rate
Compute detection probability of each node P_{D_i} by equation (6)
end
end
P_D <— compute equation (11)
Store $P_D(x, y)$ in the 2D array
Associate P_F <— compute equation (12)
return $(\mathbf{x_{min}}, \mathbf{y_{min}})$ **<—** $\boldsymbol{min_{x,y}(P_D(x_{min}, y_{min}))}$
end

The algorithm above gives the $P_{D_{min}}$ based on the current configuration and will be used in the algorithm to identify the finalized configuration later.

IV. Logistic Regression Learning Model

In this paper, a non-uniform probability distribution of fire risk across the whole geographic region is considered. In reality, the probability of having a wildfire at a specific location depends on lots of factors. Here, we consider three factors as an example: vegetarian type, month and cause. Table IV gives 100 data with labels. In order to learn a model, which has the capability of predicting the probability of having a wildfire at different locations, a logistic regression algorithm has been introduced. A key idea in the logistic regression algorithm is to minimize the in-sample-error, E_{in}, which, hopefully, could have a good generalization property. It is generally known that the error measure of logistic regression is very difficult if we directly compute the error between the prediction and the real value and minimize that error, for instance, by taking the derivative. Based on this fact,

gradient descent is introduced and is widely used to minimize the in-sample-error when learning the logistic regression model.

The error measure, normally, could be defined as a cross-entropy error:

$$E_{in}\left(\vec{w}\right) = \frac{1}{n}\sum_{i=1}^{n} ln\left(1 + e^{-y_i \sum_{j=0}^{d} w_j x_{ij}}\right) \quad (14)$$

The E_{in} defined here can be proved that it is positive semidefinite, which suggests that the E_{in} is convex. Therefore, the gradient descent method is generally considered. In such a method, two parameters need to be defined, the direction vector $\vec{v}$ and step size α. As defined under Section 1.2 Gradient Methods — Convergence in the book *Nonlinear Programming* [4], the equation is defined as,

$$x^{k+1} = x^k - \alpha^k D^k \nabla f(x^k) \quad (15)$$

In the equation, α is the step size, $D^k \nabla f(x^k)$ is the direction vector. Also, as suggested in the same section, the diminishing step size method is used. The following two expressions define the direction vector and step size used during the logistic regression training process:

$$\vec{v}_t = -\frac{\nabla_{\vec{w}} E_{in}(\vec{w}_{(t)})}{\|\nabla_{\vec{w}} E_{in}(\vec{w}_{(t)})\|} \quad (16)$$

$$\eta_t = \eta_0 \|\nabla_{\vec{w}} E_{in}(\vec{w}_t)\| \quad (17)$$

V. Learning Process

The logistic regression hypothesis, a vector $\vec{w}$, should be learned from historical data, such as the training data provided in Table IV in Appendix. The maximum iterations, tolerance, and initial step size should also be specified. After the learning, a particular hypothesis will be obtained and be applied to any new given data to predict the probability. Readers may further analyze the out-of-sample-error to qualify the generalization property, which is beyond the scope of this paper. The pseudo code is provided below:

PSEUDO CODE II

LOGISTIC REGRESSION ALGORITHM

```
begin
        Given training data set with label, iteration
        number, initial step size
        Initialize w⃗ as 0⃗, t = 0
        while t < iteration number
                Calculate v⃗_t <— equation (16)
                Update the model w⃗_{t+1} <— w⃗_t − η_t × v⃗_t
                t = t + 1
        end
        return w⃗
end
```

At this point, the deployment solution and probabilistic model has been fully defined. The performance of incorporating the additional probability predicted by the logistic regression model will be analyzed in the experiment design and simulation section. But to fully measure the performance of the modified probability model, we should also consider the whole reconfiguration phase. In the next section, a particular algorithm discussed in the original paper has been discussed. The overall computation performance has also been optimized.

CENTRALIZED COVERAGE PROTOCOL ALGORITHM

I. A Brief Review

There will be many cases where the current configuration fails to cover the geographic region. For example, active nodes will consume more energy than the inactive nodes, which results in running out of battery earlier. As the number of active nodes decreases, equation (1) will no longer be valid. Thus, a reconfiguration will be required at that moment. After running a specific algorithm in the reconfiguration phase, the sensor network can be reconfigured and, again, meet the requirement of equation (1). There can be multiple algorithms to achieve such a goal. As this paper is extended from the "Efficient Coverage Maintenance Based on Probabilistic Distributed Detection" [1], we firstly briefly go through the algorithm developed in that paper and make the improvement after then.

During the reconfiguration phase, the centralized algorithm put all nodes in an inactive set. The phase starts by randomly picking up a node from the inactive set and put into an active set, setting the local false alarm rate to be same as the passed in system false alarm rate. A loop will be executed, the algorithm computes the $P_{D_{min}}$ among all the nodes in the active set. If the $P_{D_{min}}$ is lower than the passed in parameter β, then put the node which is closest to the point with the lowest detection probability into the active set. The loop keeps running until the $P_{D_{min}}$ is above the threshold or all the inactive nodes have been put into the active set. The algorithm returns success or failure, respectively.

II. Centralized Coverage Protocol Algorithm

The pseudo code for the original Centralized Coverage Protocol algorithm is presented below:

PSEUDO CODE III

CENTRALIZED COVERAGE PROTOCOL ALGORITHM

```
begin
        Given the system false alarm rate α and detection
        probability β
        Initialize an empty inactive set and empty active set
        Iterate through all the nodes, putting all their ID
        number in the inactive set
        Randomly select a node form the inactive set and
        put it into the active set
        Set α_0 = α
        Set P_{D_min} <— run the Minimum Detection
        Probability Algorithm
        while P_{D_min} < β
                if size of inactive set == 0
                        return failure
```

(x_{min}, y_{min}) <— run the Minimum Detection Probability Algorithm
$P_{D_{min}}$ <— $P_D(x_{min}, y_{min})$
Put the node with the closest location to (x_{min}, y_{min}) and with the longest distance from all currently active nodes

end

return the active set

end

However, as we have already defined, a sensor network should, to some extends, be aware of the nature of each location. From the hypothesis learned from the logistic regression, the Centralized Coverage Protocol Algorithm should take the probability of having a wildfire for each location into consideration. Therefore, we modify Centralized Coverage Protocol Algorithm and present the pseudo code below:

PSEUDO CODE IV

MODIFIED CENTRALIZED COVERAGE PROTOCOL ALGORITHM

begin

Given the system false alarm rate α, detection probability β, training data set X, and region A
Initialize an empty inactive set and empty active set
Learn a hypothesis $\vec{w}$ <— Pseudo Code II
Initialize $P_{F_{max}} = -\infty$
Iterate through all the nodes, putting all their ID number in the inactive set
for each location (x, y) in region A
Predict the probability $p(x, y)$ <— sigmoid $(\vec{w} \times$ feature value at $(x, y))$
Calculate the likelihood ratio $\lambda(x, y)$ <— equation (9)
end
Randomly select a node from the inactive set and put it into the active set
while $P_{D_{min}} < \beta$ **and** $P_{F_{max}} < \alpha$
for each location (x, y) in region A
for each node i in region active set
Compute $P_{D_i}(x, y)$ <— equation (6)
Compute $P_{F_i}(x, y)$ <— equation (5)
Compute $P_D(x, y)$ <— equation (11)
Compute $P_F(x, y)$ <— equation (12)
$P_{D_{min}}$ <— $min(P_D(x, y)$ for all locations$)$
$P_{F_{max}}$ <— $max(P_F(x, y)$ for all locations$)$
return active set

end

The sensor network will be reconfigured by using all the nodes in the active set and schedule all the other nodes to be inactive until the next time reconfiguration phase has been called. The inactive nodes will wake every number units of time and check whether it is required to be set as an active node in the next reconfiguration phase. The centralized algorithm is enough to configure the sensor network which is going to cover the whole geographic region.

COMPUTATION COST IMPROVEMENT OF THE CENTRALIZED COVERAGE PROTOCOL ALGORITHM

I. Issue Addressing

The overall algorithm runs by adding an additional node in each iteration, recomputing $P_{D_{min}}$ by recomputing the summation of all the possible combinations of nodes' decisions which satisfy the majority rule on all possible locations. Such an algorithm will only be efficient if using a proper algorithm during the implementation. This point is very crucial. The algorithm used in the reconfiguration phase is an offline algorithm. The whole network will not be able to monitor any situation occurred during that time. Specifically, the probability of detecting a campfire given the network is in the reconfiguration phase would be zero. Therefore, the expected time spending on reconfiguring the network should be as short as possible.

The efficiency of computing P_D has to be optimized by lowering the computation cost. As we have noticed, the required computation of P_D for each location is the most expensive cost. It considers all possible combinations of nodes' decisions and take the summation of all the valid results. When applying the network in reality, a large-scale geographic region will lead to an increase of the number of nodes, as suggested under the section of sensor nodes placement, which results in the increase of the time to compute P_D . Thus, we propose three approaches in our paper, which aims to reduce the computation cost. The first two solutions give the exact solution with time complexity $O(2^n)$, and the last one gives an approximate solution but with a time complexity $O(n^2)$.

II. Algorithm to improve calculating P_D

Approach 1

A recursion algorithm has been developed. As for the recursion tree, the number of nodes defines the height of recursion tree. In each layer, each parent node will have two branches, either multiplying P_{D_i} or $(1 - P_{D_i})$. Thus, the overall time complexity is $O(2^n)$.

Approach 2

We also propose a dynamic programing (DP) solution, which also gives the exact solution but with less than $O(2^n)$ time complexity. As we have noticed, for each location, the P_D is computed as the summation of all possible combinations of nodes' decisions, pruned based on the majority rule. Therefore, each result computed will become a sub-result required to compute in other configurations.

A specific example of 5 active nodes is created to demonstrate this approach. The Table I and II in Appendix give the probability of $\prod_{i=1}^{5} P_{D_i}$, and the probability of $\prod_{i=1}^{n}(1 - P_{D_i})$. The combination of these two tables will be

enough to calculate all the probability necessary to calculate the P_D, and the result is shown in Table III in Appendix.

The second approach has a great potential to be optimized by applying appropriate parallel computing techniques. Results in the same column do not depend on each other but only on the previous column. Thus, we can compute the results in the same column, which will result in an overall increase of computation speed. But we do not say that the first approach is always inferior to the second. For example, the first approach is more interpretable and easier to implement.

Approach 3

The third approach mainly aims to increase the computation speed with a decrease of solution quality as the cost. We notice that P_D is the sum of products, so we can calculate its lower bound based on Jensen's inequality [4].

$$\begin{aligned} &ln\, P_D(x,y) \\ &= ln \sum_{\{|S_1|>|S_0|\}} \prod_{\{i\in S_o\}} \left(1 - P_{D_i}(x,y)\right) \prod_{\{j\in S_1\}} P_{D_j}(x,y) \\ &= ln \frac{\sum_{\{|S_1|>|S_0|\}} \prod_{\{i\in S_o\}} \left(1-P_{D_i}(x,y)\right) \prod_{\{j\in S_1\}} P_{D_j}(x,y)}{\sum_{j=\lfloor N/2\rfloor+1}^{N} \binom{N}{j}} + \ln \sum_{j=\lfloor N/2\rfloor+1}^{N} \binom{N}{j} \end{aligned} \quad (18)$$

If N is the number of active nodes being considered, then there are $\sum_{j=\lfloor N/2\rfloor+1}^{N} \binom{N}{j}$ possible combinations.

Now apply Jensen's inequality to the first term.

$$\begin{aligned} &ln P_D(x,y) - \ln \sum_{j=\lfloor N/2\rfloor+1}^{N} \binom{N}{j} \\ &\geq \frac{\sum_{\{|S_1|>|S_0|\}} ln\left(\prod_{\{i\in S_o\}} \left(1-P_{D_i}(x,y)\right) \prod_{\{j\in S_1\}} P_{D_j}(x,y)\right)}{\sum_{j=\lfloor N/2\rfloor+1}^{N} \binom{N}{j}} \end{aligned} \quad (19)$$

Then focus on the numerator of right-hand side.

$$\begin{aligned} &\sum_{\{|S_1|>|S_0|\}} ln\left(\prod_{\{i\in S_o\}} \left(1 - P_{D_i}(x,y)\right) \prod_{\{j\in S_1\}} P_{D_j}(x,y)\right) \\ &= \sum_{\{|S_1|>|S_0|\}} \left(\sum_{\{i\in S_0\}} ln\left(1 - P_{D_i}(x,y)\right) + \sum_{\{j\in S_1\}} ln\, P_{D_j}(x,y)\right) \end{aligned} \quad (20)$$

If we expand the sum, we see that the expression constitutes a linear combination of all the $ln\, P_{D_i}(x,y)$ and $ln\left(1 - ln\, P_{D_i}(x,y)\right)$ where *i* is indices for all the active nodes. The number of times each $ln\, P_{D_i}(x,y)$ term appears should be:

$$A = \sum_{j=\lfloor N/2\rfloor}^{N-1} \binom{N-1}{j} \quad (21)$$

The number of times each $ln\left(1 - ln\, P_{D_i}(x,y)\right)$ term appears should be:

$$B = \sum_{k=0}^{\lfloor N/2\rfloor-1} \binom{N-1}{k} \quad (22)$$

Both of these can be calculated in $O(n^2)$ time, and adding these together requires $O(n)$ time, so the total time complexity of the algorithm is $O(n^2)$.

Since given each choice of a particular sensor's vote (detection / no detection), all other possible combinations of the majority vote case can be calculated, and the number of times each sensor's each type of vote appears when calculating the system P_D can be calculated.

Thus, it is obtained that:

$$\begin{aligned} &ln\, P_D(x,y) \\ &\geq \frac{B\cdot\sum_{\{i\in S_0\}} ln\left(1-PD_i(x,y)\right) + A\cdot\sum_{\{j\in S_1\}} ln\, PD_j(x,y)\right)}{\sum_{j=\lfloor N/2\rfloor+1}^{N} \binom{N}{j}} + \ln \sum_{j=\lfloor N/2\rfloor+1}^{N} \binom{N}{j} \end{aligned} \quad (23)$$

$$\begin{aligned} &P_D(x,y) \\ &\geq exp\left(\frac{B\cdot\sum_{\{i\in S_0\}} ln\left(1-PD_i(x,y)\right) + A\cdot\sum_{\{j\in S_1\}} ln\, PD_j(x,y)\right)}{\sum_{j=\lfloor N/2\rfloor+1}^{N} \binom{N}{j}} + \ln \sum_{j=\lfloor N/2\rfloor+1}^{N} \binom{N}{j}\right) \end{aligned} \quad (24)$$

These three approaches are developed to minimize the computation cost. The first two have $O(2^n)$ time complexity, but the second could potentially be faster by using a carefully designed parallel computing algorithm. The third approach do not give an exact result but greatly decrease the computation cost. All these three approaches could be applied to the wireless sensor network to monitor the forest fire. A lower computation cost algorithm leads to a short reconfiguration time, which is normally more preferred in the real-world application.

EXPERIMENT DESIGN AND SIMULATION

Two experiments have been created in order to compare the performance between the original work and the optimization solution we present in this paper. In order to make a good comparison, we firstly reproduce the original Centralized Coverage Protocol Algorithm. This algorithm assumes a uniform wildfire occurrence probability across the entire geographic region. The original work was announced to have a $O(2^n)$ time complexity without explicitly specify which algorithm being used. The two experiments, *A* and *B*, are developed to present the improvement we have made in this paper. The first experiment is designed to show the optimized solution quality based on the modified probability model, and the second experiment is for demonstrating the computation cost optimization.

For each of the heatmaps in Figures II-IV, the color of each block represents the system P_D at each point in space, i.e. given there is an event at the block, what is the probability that the whole network, using a majority vote decision rule, would say that an event has occurred. Different heatmaps represent the P_D when different nodes turned on. Going from one heatmap to the other, the system has decided to turn on a new node, and the P_D's should be recalculated, thereby leading to a different heatmap.

I. Original Work Reproduction

In our first simulation, we recreate the result of the previous paper [1] by computing the exact P_D. The system P_D in our simulation of a 12×12 array environment with 16 sensor nodes evenly distributed in a 4×4 fashion shows the following progression (Figure II). The recursive approach (depth first search approach) is used to compute the exact P_D, which takes $O(2^n)$ time. Six nodes are turned on before the

coverage criterion$(\alpha = 0.1, \beta = 0.9$; see Equation (1)$)$ is met. Without specification, these thresholds are used throughout the following sections.

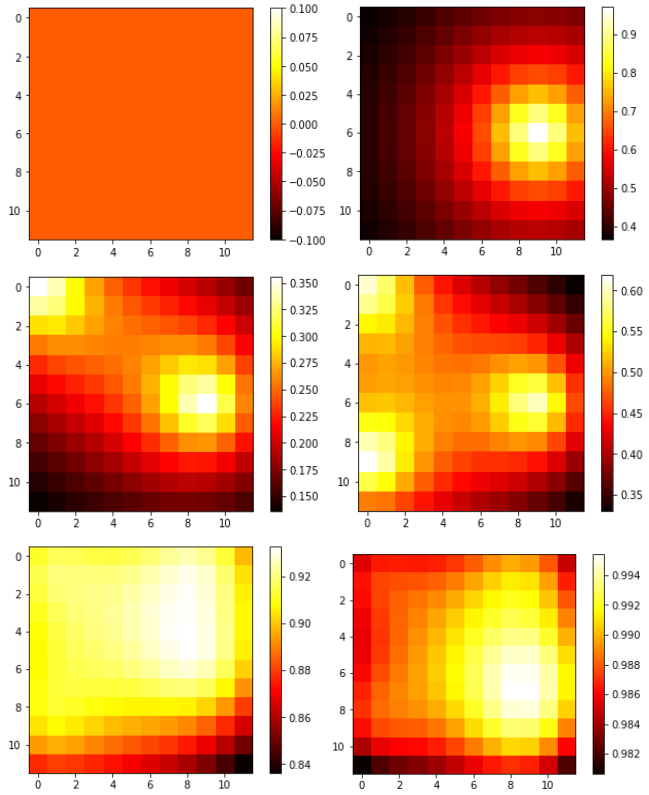


FIGURE II

HEAT MAP REPRESENTATION OF THE EVOLUTION OF SYSTEM P_D IN A 12×12 ENVIRONMENT WITH 16 (ID: 0~16) SENSORS EVENLY DISTRIBUTED. THE COLOR OF EACH BLOCK INDICATES THE SYSTEM P_D IN THAT BLOCK. AFTER TURNING ON 6 SENSORS (IDS: 11, 0, 12, 3, 15, 13; LAST ONE NOT SHOWN), THE COVERAGE CRITERION IS MET.

The algorithm converges after six nodes were turned on, which takes on average 35 seconds. After each node is turned on, there is a local increase of P_D around the new node. When new nodes are turned on, these probability masses merge to cover the whole area. A total of six sensors is sufficient to cover the whole area, and they are sparsely distributed within the area.

II. Experiment A: Logistic Regression for Non-uniform Probability Prediction

In this experiment, a training data set with three features and label is used in the training process. The data set contains 100 data points. Each data has three features. The specific value of each data point is modified from historical data. Please note the example here is only for the demonstration purpose. The learning model will be different by using different training set. In our example, the model learned is $\vec{w}$ = [0.1872, 0.1224, -0.1309, -0.0324], with 1,000,000 iterations and initial step size being 10^{-5}.

This algorithm uses logistic regression to model the prior probability of H_0 and H_1 which can be used to calculate the decision threshold. To perform Maximum A Posteriori reasoning, we must have a node activate when the inequality in Equation (9) is satisfied. It takes 25 seconds to turn on six nodes in a simulated 10×10 block environment with 16 sensors.

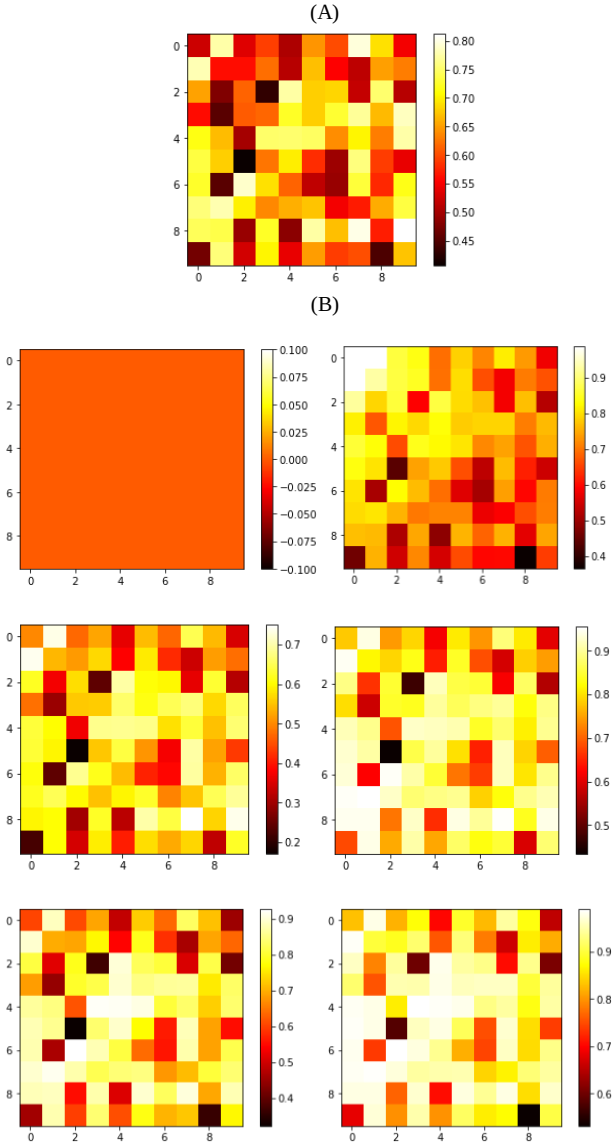


FIGURE III

(A) THE PRIOR PROBABILITY OF FIRE PREDICTED BY LOGISTIC REGRESSION. (B) THE EVOLUTION OF SYSTEM P_D IN A 10×10 ENVIRONMENT WITH 16 (ID: 0~16) SENSORS EVENLY DISTRIBUTED, AFTER TURNING ON 5 SENSORS (IDS: 0, 8, 6, 4, 3).

It is clear in the pattern in Figure III that places that have a higher prior probability of event will have a lower decision threshold, and so will in turn have higher detection probability. In other words, the system more attentively listens to any signal from high-risk areas predicted by our logistic regression model, leading to higher P_D in those areas, as well

as in adjacent areas, as can be seen in the broadening of the bright spots near high risk areas.

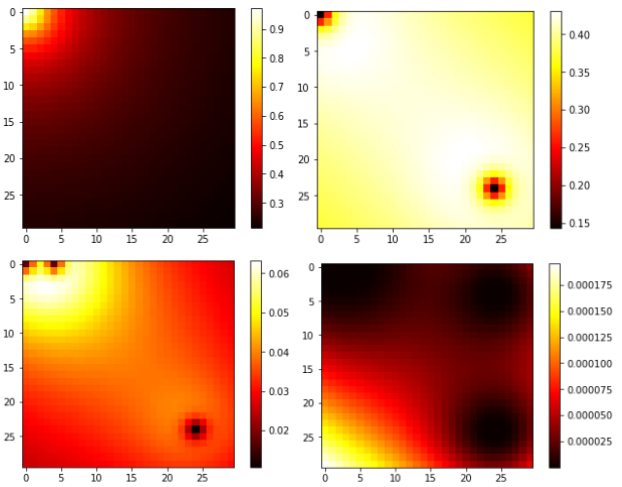
The modified probability model gives us the advantage of making use of features like weather, vegetation and terrain to compute the prior probability of a hazardous event in each area in space. Using the results from our logistic regression model, we can calculate the decision boundary for each point based on relevant geographic information, instead of assuming a uniform distribution of probability of hazardous over the whole region.

However, we note that our synthetic data is very noisy, and our simulated environment is discrete and highly coarse grained. Due to this reason, in our experiment, P_F almost always increases with each additional active node, and our objectives are not met before all nodes are turned on. In a real environment, risk for hazardous events in neighboring regions will tend to be more correlated, and the transition may be smoother.

III. Experiment B: Lowering the Computation Cost of P_D

As we have developed in previous sections, the second and third approach is able to increase the computation speed. The second approach seems to more depend on the specific parallel programing skill and will not affect the solution quality. Therefore, it is not worthy to be discussed here. However, the third solution increases the computation speed from a totally different approach but also have lost the solution quality as a compensation. Therefore, the Experiment B has been developed. The algorithm uses Jensen's inequality to calculate a lower bound of the system P_D to avoid the cumbersome exact calculation given in the original paper [1].

In an experiment of scale similar to previous experiments, the 16 sensors are evenly distributed in a 12 × 12 environment. It takes only 12 seconds for six of the nodes to activate. To demonstrate the speed of the algorithm, we simulated 49 sensors evenly distributed in an environment of 30 × 30 blocks. It takes 48 seconds for six of the nodes to activate.



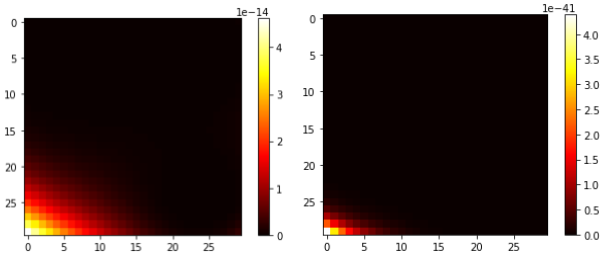


FIGURE IV

THE EVOLUTION OF P_D USING JENSEN'S INEQUALITY FOR LOWER BOUNDING. THE FOLLOWING SENSORS ARE ACTIVATED IN ORDER: 0, 48, 1, 13, 2, 8. LOOSE LOWER BOUND OF JENSEN'S INEQUALITY CAUSES UNDERFLOW.

The algorithm demonstrates similar behavior as the inefficient DFS algorithm in that it captures the ratio of P_D between different points. It has time complexity of $O(n^2)$, but it fails for larger N, since the value $\binom{N}{\lfloor N/2 \rfloor}$ grows too fast compared to the numerator. The loose lower bound [4] causes numerical underflow. A possible remedy to this is to add correction terms based on the approximation, $E[log(x)] \approx log(E[x]) - \frac{Var[x]}{2E[x]^2}$, where $x = \prod_{\{i \in S_o\}}(1 - P_{D_i}(x,y)) \prod_{\{j \in S_1\}} P_{D_j}(x,y)$.

To be able to calculate or approximate the mean and variance of x efficiently, this approach requires either strong assumptions about the form of the distribution of $\prod_{\{i \in S_o\}}(1 - P_{D_i}(x,y)) \prod_{\{j \in S_1\}} P_{D_j}(x,y)$, or some numerical approximations of the distribution. The necessary number of nodes in an area can be approximated by running the inefficient DFS algorithm for representative terrains in the area. When deploying the sensor network online, the advantage of the current approach in rapid reconfiguration can be exploited, and after a certain number of nodes is activated, the system can terminate.

IV. Experiment Summary

As Experiment *A* and *B* sufficiently show that the original problem has been optimized as we expected. By applying the probability distribution predicted by the logistic regression model, the wireless sensor network is able to "sense" the nature of each location and optimize the solution quality by considering this additional factor. From the Experiment *B*, even though the final solution quality decrease, the algorithm's runtime speed is significantly improved.

FUTURE WORK

The current paper attempts to solve two problems: the incorporation of regression techniques to predict high risk areas, and the approximation of P_D, which would otherwise require $O(2^n)$ time to compute.

Future work should focus on how to make our algorithm more scalable. This involves first developing more accurate approximation techniques for P_D.

Fusion centers can also be included, each of which interact with only a local subset of sensor nodes. The fusion centers can communicate with other fusion centers about the coverage in its own territory, as well as warn neighboring nodes if an event has been detected.

The sensor network should also be able to update its decision threshold while monitoring of the environment. Risk factors such as temperature, humidity and human activity can be incorporated to calculate the decision threshold in real time. Areas that show higher risk demand sharper detection, i.e. lower decision threshold.

Our system mainly focuses on the prevention of an event – how to detect an event as soon as it starts. Another interesting topic is how to track the trajectory or range of an event. When there is a change in the risk evaluation or even when an actual event is detected, the system should be able to propagate this warning message to neighboring nodes. Based on the speed of propagation of the activation of nodes, the speed at which the event is spreading can be estimated. More efficient containment of the event would be possible if it is possible to accurately plan the construction of barriers based on the spread of, for example, a forest fire.

CONCLUSION

Extending from previous work, this paper makes a series of modifications to make the original probability model applicable to monitor the real-world forest wildfire. Two important optimizations have been made. The logistic regression is introduced to learn a model, which is for determining a probability distribution. The modified Centralized Coverage Protocol Algorithm is capable of activating nodes by considering this additional probability distribution. In the reconfiguration phase, both parallel DP solution and Jensen's inequality effectively reduce the work required for computing the detection probability at each location. Admittedly, these two optimizations could be further improved. The probability distribution from the logistic model will dominate the detection probability, which leads to a failure in the reconfiguration phase; and the Jensen's inequality has a loose bound, resulting in a numerical underflow. However, the examples demonstrated in this paper have shown the promising potential of our optimization to the real-world application. A coverage maintenance protocol considering the wildfire occurrence probability with a short reconfiguration time is always desirable.

CONTRIBUTION

Jianxing Liao:
Depth first search coding and implementation
Jensen's inequality coding
Logistic regression analysis
Reproduction of previous work coding

Chong Wang:
Dynamic programming table
Incorporating and coding logistic regression into our model
Jensen's inequality derivation
Experiment A analysis

Xinyu Zhang:
Experiment B analysis
Jensen's inequality analysis
Reproduction of previous work coding
Logistic regression coding

Together:
Model derivation
Formula derivation
Pseudo-code
Article review
Resource search
Future work
Report write-up

REFERENCES

[1] Xing, Guoliang, et al. "Efficient coverage maintenance based on probabilistic distributed detection." *IEEE Transactions on Mobile Computing* 9.9 (2010): 1346-1360.

[2] Hefeeda, Mohamed, and Majid Bagheri. "Randomized k-Coverage Algorithms For Dense Sensor Networks." *INFOCOM*. 2007.

[3] Manolakos, Elias S., Dimitrios V. Manatakis, and Gavriil Xanthopoulos. "Temperature field modeling and simulation of wireless sensor network behavior during a spreading wildfire." *Signal Processing Conference, 2008 16th European*. IEEE, 2008.

[4] Gao, Xiang, Meera Sitharam, and Adrian E. Roitberg. "Bounds on the Jensen Gap, and Implications for Mean-Concentrated Distributions." *arXiv preprint arXiv:1712.05267* (2017).

[5] Bertsekas, Dimitri P. *Nonlinear programming*. Belmont: Athena scientific, 1999.

APPENDIX

TABLE I

ALL COMBINATIONS OF $\prod_{i=1}^{N} P_{D_i}$

	1	2	3	4	5
A	P_{D1}	P_{D2}	P_{D3}	P_{D4}	P_{D5}
B		$P_{D1}P_{D2}$ = (1A) P_{D2}	$P_{D1}P_{D3}$ = (1A) P_{D3}	$P_{D1}P_{D4}$ = (1A) P_{D4}	$P_{D1}P_{D5}$ = (1A) P_{D5}
C			$P_{D2}P_{D3}$ = (2A) P_{D3}	$P_{D2}P_{D4}$ = (2A) P_{D4}	$P_{D2}P_{D5}$ = (2A) P_{D5}
D			$P_{D1}P_{D2}P_{D3}$ = (2B) P_{D3}	$P_{D3}P_{D4}$ = (3A) P_{D4}	$P_{D3}P_{D5}$ = (3A) P_{D5}
E				$P_{D1}P_{D2}P_{D4}$ = (2B) P_{D4}	$P_{D4}P_{D5}$ = (4A) P_{D5}
F				$P_{D1}P_{D3}P_{D4}$ = (3B) P_{D4}	$P_{D1}P_{D2}P_{D5}$ = (2B) P_{D5}
G				$P_{D2}P_{D3}P_{D4}$ = (3C) P_{D4}	$P_{D1}P_{D3}P_{D5}$ = (3B) P_{D5}
H				$P_{D1}P_{D2}P_{D3}$ P_{D4}= (3D) P_{D4}	$P_{D1}P_{D4}P_{D5}$ = (4B) P_{D5}
I					$P_{D2}P_{D3}P_{D5}$ = (3C) P_{D5}
J					$P_{D2}P_{D4}P_{D5}$ = (4C) P_{D5}
K					$P_{D3}P_{D4}P_{D5}$ = (4D) P_{D5}
L					$P_{D1}P_{D2}P_{D3}$ P_{D5}= (3D) P_{D5}
M					$P_{D1}P_{D2}P_{D4}$ P_{D5}= (4E) P_{D5}
N					$P_{D1}P_{D3}P_{D4}$ P_{D5}= (4F) P_{D5}
O					$P_{D2}P_{D3}P_{D4}$ P_{D5}= (4G) P_{D5}
P					$P_{D1}P_{D2}P_{D3}P_{D4}$ P_{D5}=(4H)P_{D5}

TABLE II

ALL COMBINATIONS OF $\prod_{i=1}^{N}(1-P_{D_i})$

	1	2	3	4	5
A	1 - P_{D1}	1 - P_{D2}	1 – P_{D3}	1 - P_{D4}	1- P_{D5}
B		(1A) (1 - P_{D2})	(1A) (1 -P_{D3})	(1A) (1 -P_{D4})	(1A) (1 -P_{D5})
C			(2A) (1 -P_{D3})	(2A) (1 -P_{D4})	(2A) (1 -P_{D5})
D			(2B) (1 -P_{D3})	(3A) (1 -P_{D4})	(3A) (1 -P_{D5})
E				(2B) (1 -P_{D4})	(4A) (1 -P_{D5})
F				(3B) (1 -P_{D4})	(2B) (1 -P_{D5})
G				(3C) (1 -P_{D4})	(3B) (1 -P_{D5})
H				(3D) (1 -P_{D4})	(4B) (1 -P_{D5})
I					(3C) (1 -P_{D5})
J					(4C) (1 -P_{D5})
K					(4D) (1 -P_{D5})
L					(3D) (1 -P_{D5})
M					(4E) (1 -P_{D5})
N					(4F) (1 -P_{D5})
O					(4G) (1 -P_{D5})
P					(4H) (1 -P_{D5})

In Table I and II, each column represents a unique combination with the current node, which helps to generate the full possibility distribution.

TABLE III

CALCULATION OF P_D

	5
A	(1- P_{D5}) P_{D1} $P_{D2}P_{D3}P_{D4}$ = (Table II 5A) (Table I 4 H)
B	(1- P_{D5}) (1- P_{D1}) $P_{D2}P_{D3}P_{D4}$ = (Table II 5B) (Table I 4 G)
C	(1- P_{D5}) (1- P_{D2}) $P_{D1}P_{D3}P_{D4}$ = (Table II 5C) (Table I 4 F)
D	(1- P_{D5}) (1- P_{D3}) $P_{D1}P_{D2}P_{D4}$ = (Table II 5D) (Table I 4 E)
E	(1- P_{D5}) (1- P_{D4}) P_{D1} $P_{D2}P_{D3}$ = (Table II 5E) (Table I 3D)
F	(1 – P_{D3}) (1 – P_{D4}) $P_{D1}P_{D2}P_{D5}$ = (Table II 4D) (Table I 5F)
G	(1 – P_{D2}) (1 – P_{D4}) $P_{D1}P_{D3}P_{D5}$ = (Table II 4C) (Table I 5G)
H	(1 – P_{D2}) (1 – P_{D3}) $P_{D1}P_{D4}P_{D5}$ = (Table II 3C) (Table I 5H)
I	(1 – P_{D1}) (1 – P_{D4}) $P_{D2}P_{D3}P_{D5}$ = (Table II 4B) (Table I 5I)
J	(1 – P_{D1}) (1 – P_{D3}) $P_{D2}P_{D4}P_{D5}$ = (Table II 3B) (Table I 5J)
K	(1 – P_{D1}) (1 – P_{D2}) $P_{D3}P_{D4}P_{D5}$ = (Table II 2B) (Table I 5K)
L	(1 – P_{D4}) $P_{D1}P_{D2}P_{D3}$ P_{D5}= (Table II 4A) (Table I 5L)
M	(1 – P_{D3}) $P_{D1}P_{D2}P_{D4}$ P_{D5}= (Table II 3A) (Table I 5M)
N	(1 – P_{D2}) $P_{D1}P_{D3}P_{D4}$ P_{D5}= (Table II 2A) (Table I 5N)
O	(1 – P_{D1}) $P_{D2}P_{D3}P_{D4}$ P_{D5}= (Table II 1A) (Table I 5O)

P	$P_{D1}P_{D2}P_{D3}P_{D4}P_{D5}$ = (Table I 5P)

P_D <— SUM(ALL ENTRIES IN TABLE III)

In Table III, each column represents the probability under the current nodes' decisions, the whole table gives all the possible combination of five nodes' decisions.

TABLE IV
LOGISTIC REGRESSION TRAINING SET

No.	Month	Vegetation	Cause	Label	No.	Month	Vegetation	Cause	Label
1	9	2	9	1	**51**	8	2	0	1
2	6	2	9	1	**52**	2	3	10	0
3	6	1	7	1	**53**	6	1	0	1
4	6	2	0	1	**54**	2	3	5	0
5	7	2	0	1	**55**	11	1	7	0
6	8	1	0	1	**56**	9	1	0	1
7	7	3	7	0	**57**	6	1	0	1
8	9	2	1	0	**58**	8	2	0	1
9	12	2	0	0	**59**	7	1	0	1
10	8	1	0	1	**60**	12	1	6	0
11	6	2	0	1	**61**	9	3	3	0
12	12	1	1	0	**62**	8	3	0	1
13	10	3	0	1	**63**	8	2	0	1
14	5	2	0	1	**64**	7	3	0	1
15	8	2	0	0	**65**	9	3	0	1
16	9	2	7	0	**66**	6	1	0	1
17	9	3	0	1	**67**	3	1	7	0
18	2	3	0	0	**68**	8	2	10	1
19	7	2	0	1	**69**	9	3	10	1
20	7	1	0	1	**70**	10	1	10	1
21	9	1	2	0	**71**	2	1	10	0
22	9	1	2	1	**72**	6	1	0	1
23	8	3	7	1	**73**	2	3	6	0
24	9	3	0	1	**74**	7	3	8	0
25	6	1	0	1	**75**	12	2	0	0
26	6	3	0	1	**76**	8	1	10	1
27	1	1	7	0	**77**	11	2	0	0
28	9	3	4	0	**78**	6	2	0	1
29	6	3	10	1	**79**	10	1	10	1
30	11	2	8	0	**80**	9	1	0	1
31	6	1	0	1	**81**	8	1	8	1
32	12	3	0	0	**82**	7	2	8	1
33	10	3	0	1	**83**	5	2	3	1
34	8	2	9	1	**84**	7	2	5	1
35	8	1	0	1	**85**	5	3	4	0
36	8	2	0	1	**86**	7	3	4	0
37	2	3	4	0	**87**	8	1	8	1
38	8	2	1	1	**88**	8	2	6	1
39	9	1	0	1	**89**	9	2	4	1
40	6	2	9	1	**90**	12	2	3	0
41	11	3	10	1	**91**	6	2	6	1
42	8	1	9	1	**92**	7	3	6	1
43	8	3	0	1	**93**	3	1	8	0
44	6	1	7	1	**94**	7	1	5	1
45	8	1	6	0	**95**	9	3	6	1
46	10	1	1	0	**96**	8	2	8	1
47	8	1	0	1	**97**	9	3	8	0
48	6	1	0	1	**98**	7	1	4	1
49	8	3	0	1	**99**	6	2	7	1
50	9	1	9	1	**100**	7	1	8	1

Table IV contains 100 training data that are used to predict wildfire. The data set comes from California Department of Forestry and Fire Protection, and we modify the data set, so that it can be used for logistic regression. For each data point, there are three features (month, vegetation and cause) and one label. The label value of 1 represents the occurrence of wildfire while the label value of 0 represents that there is no fire. For the month feature, the values represent the specific month, so the values

are between 1 and 12, inclusive. For the vegetation feature, 1 represents brush, 2 represents grass and 3 represents timber. For the cause feature, 0 represents undetermined wildfire cause, 1 represents wildfire caused by playing with fire, 2 represents wildfire caused by railroad, 3 represents wildfire caused by debris burning, 4 represents wildfire caused by equipment, 5 represents wildfire caused by campfire, 6 represents wildfire caused by electrical power, 7 represents wildfire caused by human, 8 represents wildfire caused by arson, 9 represents wildfire caused by vehicle and 10 represents wildfire caused by lightning.

TABLE V

PREDICTION BASED ON HYPOTHESIS LEARNED BY LOGISTIC REGRESSION

No.	Month	Vegetation	Cause	Calculated Probability	No.	Month	Vegetation	Cause	Calculated Probability
1	3	2	6	0.5246	**51**	10	2	4	0.735
2	11	2	1	0.7756	**52**	7	2	1	0.6792
3	5	3	8	0.5368	**53**	1	3	9	0.4074
4	5	2	5	0.5928	**54**	4	1	1	0.6257
5	4	3	8	0.5063	**55**	8	2	2	0.6985
6	6	1	6	0.6449	**56**	7	3	10	0.5812
7	5	1	8	0.6009	**57**	1	1	6	0.4961
8	12	2	1	0.7962	**58**	10	2	0	0.7595
9	9	2	7	0.6901	**59**	6	3	5	0.5908
10	3	2	3	0.5487	**60**	2	2	0	0.5425
11	11	2	0	0.7811	**61**	8	1	2	0.7253
12	4	3	1	0.5627	**62**	1	2	7	0.4554
13	4	1	9	0.5633	**63**	12	1	6	0.791
14	6	3	1	0.6217	**64**	8	2	3	0.6916
15	1	2	0	0.512	**65**	6	2	6	0.6144
16	8	3	2	0.6703	**66**	3	2	7	0.5165
17	4	2	6	0.555	**67**	2	3	2	0.4937
18	3	3	3	0.5162	**68**	12	3	8	0.732
19	6	2	1	0.652	**69**	4	1	7	0.5791
20	5	2	0	0.6313	**70**	9	1	7	0.7174
21	6	1	5	0.6523	**71**	12	3	4	0.7566
22	1	1	8	0.4799	**72**	12	3	0	0.7797
23	6	1	10	0.6147	**73**	11	3	9	0.7005
24	1	2	10	0.4314	**74**	6	2	3	0.6371
25	12	1	9	0.7745	**75**	5	1	0	0.6612
26	7	2	1	0.6792	**76**	6	1	2	0.674
27	8	3	0	0.6844	**77**	3	2	3	0.5487
28	5	3	10	0.5207	**78**	4	1	8	0.5712
29	11	3	1	0.752	**79**	6	1	4	0.6596
30	1	1	4	0.5123	**80**	10	2	4	0.735
31	5	3	5	0.5609	**81**	12	3	6	0.7445
32	1	3	3	0.4551	**82**	12	3	7	0.7383
33	4	1	3	0.6104	**83**	2	2	6	0.494
34	5	1	6	0.6164	**84**	9	2	2	0.7237
35	9	1	6	0.7239	**85**	2	3	3	0.4856
36	7	1	5	0.6795	**86**	12	2	5	0.7743
37	10	1	9	0.7289	**87**	8	1	10	0.6708
38	11	2	2	0.7699	**88**	12	1	4	0.8016
39	10	3	10	0.6671	**89**	3	1	4	0.573
40	12	1	7	0.7856	**90**	12	1	2	0.8117
41	10	2	7	0.7157	**91**	1	1	9	0.4718
42	9	2	10	0.6689	**92**	12	2	8	0.7569
43	2	3	1	0.5018	**93**	1	1	2	0.5285
44	12	3	5	0.7506	**94**	9	3	1	0.7035
45	11	2	5	0.7522	**95**	2	2	0	0.5425
46	10	1	6	0.7477	**96**	7	3	1	0.6501
47	8	3	6	0.641	**97**	6	2	9	0.5911
48	9	3	1	0.7035	**98**	4	1	4	0.6026
49	5	1	4	0.6316	**99**	1	2	8	0.4474
50	11	2	1	0.7756	**100**	6	1	2	0.674

In Table V, a data set of size 100 is used to simulate 100 locations. Each location has three features. By applying the model learned from the logistic regression, each location now is associate with a probability, indicating the probability of wildfire occurrence. The whole table will be the full probability distribution, which will be considered when calculating the detection probability at each location.