Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

TOWARDS APPLICATIONS OF PARTICLE FILTERS IN WILDFIRE SPREAD SIMULATION

Feng Gu

Department of Computer Science
Georgia State University
34 Peachtree Street, Suite 1450, Atlanta, GA 30303, USA

Xiaolin Hu

Department of Computer Science
Georgia State University
34 Peachtree Street, Suite 1450, Atlanta, GA 30303, USA

ABSTRACT

Wildfire propagation is a complex process influenced by many factors. Simulation models of wildfire spread, such as DEVS-FIRE, are important tools for studying fire behavior. This paper presents how the sequential Monte Carlo methods, i.e., particle filters, can work together with DEVS-FIRE for better simulation and prediction of wildfire. We define an application framework of particle filters for the problem of wildfire spread using the DEVS-FIRE model, and discuss several applications. A case study example is provided and preliminary results are presented.

1 INTRODUCTION

Wildfire propagation is a complex process influenced by many factors, such as spatial fuel, weather conditions, and landscape. In past years, several models were developed to study and predict the forest fire spread. Examples include FARSITE (Finney 1998), BehavePlus (Andrews et al. 2005), and DEVS-FIRE (Natimo et al. 2008). In all these models, the states of the system, such as fire front, rate of spread speed, perimeter, and burned area at different time steps are computed based on information including spatial fuel data, weather data, and landscape data. In a dynamic data-driven environment where timely sensor data (e.g., fire fronts and burned areas of a wildfire from satellite images) are available, it is desirable to update a simulation model with the current system state for better simulation and prediction of wildfire. The problem arises about state estimation of dynamic systems.

Sequential Monte Carlo methods, often referred to as particle filters, are a set of statistical methods to study dynamic systems by recursively estimating the probability density function that can be used to compute different estimated states (Doucet et al. 2001; Schön 2006). In practical applications, the probability density function is represented by a set of samples, also called particles and their corresponding weights. Many approaches of generating samples are proposed to solve the problem that we cannot obtain the samples from the target density. These include perfect sampling, sampling importance resampling, acceptance-rejection sampling, Metropolis-Hastings independence sampling. Each of these techniques has its own advantages and disadvantages, so they can be adopted in various occasions (Simandl et al. 2007).

In this paper, we formalize the application of particle filters based on the wildfire spread model of DEVS-FIRE (Natimo et al. 2008). Within this framework, we define several applications of particle filters that could be developed using the wildfire spread simulation model. These applications include dynamic state estimation, static parameter calibration, and reconstruction of historical timely parameters. As a case study example, we show how the dynamically changing wind speed and wind direction in a fire spreading scenario can be estimated based on observed fire fronts and burned areas using the DEVS-FIRE model. We note that although the case study example deals with wind data, the formalized particle filters-based framework can be adapted and extended to other applications for wildfire spread.

The rest of this paper is organized as follows. Section 2 briefly discusses related applications of particle filters. Section 3 describes the basic methods and algorithms of particle filters. Section 4 formalizes the applications of particle filters in DEVS-FIRE. Section 5 provides a case study example. Section 6 presents the experiment results and analysis. Section 7 draws some conclusions.

2 RELATED WORK

Particle filters have been used in many applications, such as image processing, communications, chemistry, biology, and social sciences. (Wang et al. 2002) used particle filters to solve the problem of multiuser detection in signal processing. (Gustafsson et al. 2002) designed a framework for the problems of positioning, navigation, and tracking based on particle filters, and some general algorithms were described. Based on a Monte Carlo method, (Fox et al. 2001) discussed robot localization, one of the key problems of mobile robots, in which the initial position of the robot was unknown. Therefore, sensor data were utilized to estimate the robot's positions. (Zhang et al. 2003) provided another application of particle filters in biology. In this paper, populations of compact long chain off-lattice polymers were generated based on a sequential Monte Carlo method to explore the relationship between packing density and chain length.

In the forest fire literature, (Bradley 2007) presented an approach to estimate and track forest fires based on particle filters in video processing, in which images information obtained from miniature air vehicles was used. Another research direction of particle filters in forest fire spread system is data assimilation. (Mandel et al. 2007; Douglas et al. 2006; Coen et al. 2007) used particle filters to study fire behaviors of wildfire models. Their work was based on two kinds of fire models, reaction-diffusion-convection partial differential equations-based model and the level set method model. By estimating the temperature and fuel supply of each cell of the area, the outputs of the fire could be calculated by the function of the fire model based on level set method. This output was treated as the measurement to update the temperature and fuel supply by comparing estimated outputs and observed data. More details can be found in these papers.

3 THE NON-LINEAR DYNAMIC SYSTEM, ITS STATE ESTIMATION, AND PARTICLE FILTERS

3.1 The Non-linear State Space Model

A typical non-linear state space model can be denoted by formula (Jazwinski 1970)

$$\begin{cases} s_{t+1} = f(s_t, \theta, t) + v_t, \\ m_t = g(s_t, \theta, t) + w_t, \end{cases} \tag{1}$$

where s_t and m_t are the state variable and the measurement variable respectively; the functions of f and g define the evolution of the state variable and the measurement variable; v_t and w_t are two independent random variables to generate the state noise and the measurement noise; θ is a set of static parameters.

According to Markov property, a system can be describe as

$$s_{t+1} \sim p_\theta(s_{t+1} | s_1, s_2, ..., s_t) = p_\theta(s_{t+1} | s_t), \tag{2}$$

where $p_\theta(s)$ describes a set of probability density functions, and $p_\theta(s_{t+1}|s_t)$ represents the evolution of the system over time $t+1$. According to the model, we can calculate the next state using the current information at time t.

Using the Markov property and hidden Markov model (Doucet et al. 2000), (1) can be changed in the form of

$$\begin{cases} s_{t+1} \sim p_\theta(s_{t+1}|s_t) = p_{v_t}(s_{t+1} - f(s_t, \theta, t)), \\ m_t \sim p_\theta(m_t|s_t) = p_{w_t}(m_t - g(s_t, \theta, t)), \end{cases} \tag{3}$$

where we use mutual independence of the measurement noise w_t to denote that of observation over t, and the process noise v_t is also mutually independent over time.

3.2 State Estimation and Filter Density

After formalizing the dynamic system as a state space model, we can estimate the future state according to the history information contained in the current state. This also comes from probability density function $p(y_t|x_s)$. For the estimation problem, t is the next or the last step of s. Similarly, it is the filtering density problem when s equals t. Considering (3), according to Bayes' theorem, we obtain the Chapman-Kolmogorov equation (Jazwinski 1970),

$$p_\theta(s_{t+1}|M_t) = \int p_\theta(s_{t+1}|s_t) p_\theta(s_t|M_t) ds_t, \tag{4}$$

$$p_\theta(s_t|M_T) = p_\theta(s_t|M_t) \int \frac{p_\theta(s_{t+1}|s_t) p_\theta(s_{t+1}|M_T)}{p_\theta(s_{t+1}|M_t)} ds_{t+1}, \tag{5}$$

and

$$p_\theta(s_t|M_t) = \frac{p_\theta(m_t|s_t) p_\theta(s_t|M_{t-1})}{p_\theta(m_t|M_{t-1})}. \tag{6}$$

From the above equations, we can see there are not effective methods to recursively estimate the states of nonlinear dynamic systems. This results in many approximation algorithms to solve the problem. These algorithms either linearize the model to solve it, or find an optimal solution by numerical methods. The widely used Kalman filter (see e.g., (Kailath et al. 2000)) belongs to the first category. Sequential Mento Carlo methods, or particle filters, are in a different category, in which probability density functions are approximated by a series of particles.

3.3 Particle Filters

Generally speaking, a particle filters algorithm is a numerical method to approximate conditional filtering distribution. Considering filter density in (6), we can obtain

$$p_\theta(s_t|M_t) \propto p_\theta(m_t|s_t) p_\theta(s_t|M_{t-1}). \tag{7}$$

Let the target density $t(s_t) = p_\theta(s_t|M_t)$, the importance weight $q(s_t) = p_\theta(m_t|s_t)$, and the sample density $sd(s_t) = p_\theta(s_t|M_{t-1})$, then we can generate the formula $t(s_t) \propto q(s_t)sd(s_t)$. Therefore, we need to obtain samples from the target density. There are various algorithms designed for this goal, including perfect sampling, importance sampling, acceptance-rejection sampling, and Metropolis-Hastings independence sampling, which were discussed in (Liu et al. 1998; Bølviken et al. 2001; Chib et al. 1995). Among all these sampling algorithms, SIR (Sampling Importance Resampling) is widely used in many applications to generate samples from the target density.

We can construct the approximation of the target density

$$\hat{t}_N(x) = \sum_{i=1}^{N} \tilde{q}(x^{[i]})\delta(x - x^{[i]}),$$

where $\tilde{q}(x^{[i]})$ is the normalized importance weight of the sample, and δ is the Delta function. The importance weight of a sample specifies the possibility of its being generated. However, this approach has the limitation that the whole process relies on these initially generated samples. To improve the algorithm, a resampling step is added according to

$$P(\tilde{x}^{[i]} = x^{[j]}) = \tilde{q}(x^{[j]}),\ i=1, 2, \dots, N.$$

More details about the algorithm can be found in (Gordon et al. 1993).

In summary, the major steps of particle filters based on sampling importance resampling is described below.

Step 1: initialize *N* particles.
Step 2: calculate importance weights.
Step 3: normalize importance weights.
Step 4: resampling.
Step 5: predict new particles for future use.
Step 6: go to Step 2 to execute the next time step.

4 APPLICATIONS OF PARTICLE FILTERS USING THE DEVS-FIRE SIMULATION MODEL

4.1 Overview of The DEVS-FIRE Model

DEVS-FIRE is an integrated simulation environment for surface wildfire spread and containment based on Discrete Event System Specification (DEVS) (Zeigler et al. 2000). It uses a cellular space to model a forest and each cell corresponds to a sub-area of the forest. Fire spreading is a propagation process that burning cells ignite their unburned neighbor cells. The speed of fire spreading is calculated based on conditions including spatial fuel data, landscape data, and weather data of the area. Besides fire spread simulation, DEVS-FIRE also supports fire suppression simulation, and optimization of firefighting resource deployment for containing a fire. More details about DEVS-FIRE can be found in (Natimo et al. 2008). In the paper, we only consider the aspect of fire spread simulation.

Ignoring the implementation details, the DEVS-FIRE fire spread model can be describe as

$$x_{t+1} = DF(fuel, aspect, slope, weather, x_t),$$

where x_t and x_{t+1} are the system states that represent the fire spreading situation (i.e., which cells are ignited and which cells are not) at time step t and $t+1$ respectively, each of which can be denoted by a two-dimensional matrix with elements 1 or 0 representing whether the cell is ignited or not. $fuel$ is a static input parameter to define the fuel models of each cell in the cell space. $slope$ and $aspect$ refer to the static input parameters to specify landscape of each cell in the cell space. $weather$ means an input parameter to define the wind speeds and wind directions at different time. DF stands for the model of the dynamic fire spread system. According to x_t, we can compute outputs of the model, such as fire fronts, fire perimeters, and burned areas.

4.2 Towards A Framework of Applications of Particle Filters Using DEVS-FIRE

Given that DEVS-FIRE is a simulation model for dynamical wildfire spread in a forest, this section discusses several applications of particle filters that could be developed based on DEVS-FIRE. These discussions intend to define a framework of the applications of particle filters to the problem of wildfire spread using the DEVS-FIRE model.

The first application corresponds to state estimation of wildfire spread using DEVS-FIRE. When we have observed real data at some time steps from sensors or satellite images, we can estimate the states, e.g., fire fronts, of the system based on the observed data using particle filters. At every time step, we generate particles, e.g., different fire front shapes, and then update the weight of each particle based on the sensor data from the real forest. According to the particles and their corresponding weights, the next fire shape can be obtained. In this way, the information from real data is incorporated into the DEVS-FIRE model to better predict the fire propagation in practical applications.

Another application of particle filters is to calibrate the static parameters of DEVS-FIRE. In this case, the static parameters should be extended with states, thus transforming the problem into an optimal filter problem (Doucet et al. 2003). For example, an important parameter in DEVS-FIRE is the *fireline intensity threshold* that is used to decide whether an ignited cell is burnable or not. Given the real fire spreading observation data, e.g., burned areas, for some time period, the threshold can be approximated by particle filters. To do that, we randomly assign *N* particles with values in a pre-defined range. Using these particles as inputs, we run DEVS-FIRE model *N* times for this time period to obtain their corresponding burned areas. Then comparing these areas with the real burned areas, we can compute the weight of each particle, and choose the particles having larger weights for use of the next step. Recursively executing this process, the threshold will converge to a small range, so we can use this as the estimated threshold.

Another application is to reconstruct timely parameters (i.e., estimate what happened before) for a given set of observation data. For example, in DEVS-FIRE, wind data always change with time advances. If knowing the fire fronts and burned areas from time 0 to $t+1$, we can estimate the past wind data conditions using particle filters. This problem is similar to that of smoothing all past states using measurement data (see (Gibson 2003) for more details).

Each of the above applications deals with a different aspect. Thus the specific details of particle filters will also be different. Despite that, they all share a similar implementation structure that corresponds to the general structure of particle filters and uses the simulation model of DEVS-FIRE. Figure 1 illustrates a structure of particle filters based on DEVS-FIRE for the state estimation application described above. Every time step, we use the particle filters component to generate particles, run simulations based on the particles, and obtain measurements. Comparing the measurements with the observed data OBD_t, we can calculate the weights of the particles, and normalize them. Using the particles and their weights, we can get the value of state variable SV_t at time t, and the estimation of the next state SV_{t+1} at time $t+1$ according to the DEVS-FIRE simulation model.

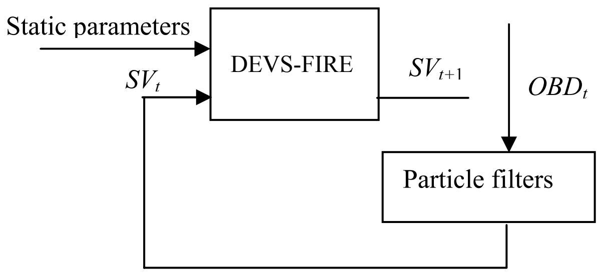

Figure 1: State estimation using DEVS-FIRE

5 A CASE STUDY EXAMPLE

5.1 The Case Study Example

As a case study example, this section shows how particle filters can be applied to wind speed and wind direction estimation using the DEVS-FIRE model. It is known that wind speed and direction can significantly affect the spreading of a wildfire. Being able to estimate the dynamic changing wind speed and direction thus is helpful and can complement the weather data collected from weather stations for better prediction of wildfire spread. In this example, we use three terms to differentiate three types of wind data: 1) *real data*– the "real" wind speeds/directions that need to be estimated. In the example, these data are artificially generated by a computer program every 10 minutes. A fire spread simulation using the real data is referred to as the "*real fire spread*"; 2) *weather data*– the wind speeds/directions that are obtained from a local weather state. These data are only available every 1 hour (assuming no measurement errors); 3) *estimated data*– the wind speeds/directions estimated from the particle filters. This example estimates the wind speeds/directions every 10 minutes. Our goal is to show that the estimated data will follow the same trend of the real data, and fire spread simulation using the estimated data gives better prediction result than using the weather data.

To apply particle filters, we define the wind data as the state variable, and estimate them according to the observation data. The observation data used in this example are the fire fronts and burned areas that are collected every 10 minutes from the real fire spread (fire spread simulation using the real data). Based on the above discussion, the dynamic system under study is defined as follows.

$$\begin{cases} wind_{t+1} = f(wind_t) + v_t, \\ B_t = calA(DF(fuel, aspect, slope, wind_t)) + w_t, \\ B_t = A_{t+1} - A_t, \end{cases}$$

where $wind_t = [wsp_t, wdir_t]$ is the state variable, the wind data (wind speed wsp_t and wind direction $wdir_t$) at time t; B_t is the measurement variable, the new burned area of the fire from time t to $t+1$; A_t and A_{t+1} are observed burned areas of the fire at time t and $t+1$ respectively. v_t and w_t are the noises of $wind_t$ and B_t respectively. $calA$ is used to calculate the burned areas from time t to $t+1$ based on the observed fire fronts at time t. DF is the DEVS-FIRE simulator, and $fuel$, $aspect$, and $slope$ are treated as the static input parameters. Here the key idea is to estimate the wind data at time t based on the new burned area after one time step for a given fire front at time t.

In the process of state estimation, the noises of the state variable and the measurement variable are used. The noises are mutually independent variables generated by normal distributions as follows.

$$v_t \sim N(0, {\sigma_v}^2), \text{ and } w_t \sim N(0, {\sigma_w}^2).$$

5.2 Implementations of Particle Filters

Based on the dynamic system formalized above, we implement the particle filter algorithm as shown below.

```
1. Initialize $N$ particles
for $i = 0$ to $N-1$
    Randomly generate $vwsp(i,0)$ and $vwdir(i,0)$
    according to normal distributions $N(0, {\sigma_{vwsp}}^2)$ and
    $N(0, {\sigma_{vwdir}}^2)$ respectively;
    $wsp(i,0) = wsp_0 + vwsp(i,0);$
    $wdir(i,0) = wdir_0 + vwdir(i,0);$
2. Compute weights
for $i = 0$ to $N-1$
    Randomly generate $vwsp(i,k)$ and
    $vwdir(i,k)$ according to normal distributions
    $N(0, {\sigma_{vwsp}}^2)$ and $N(0, {\sigma_{vwdir}}^2)$ respectively;
    $wsp(i,k) = f_{wsp}(wsp(i,k-1)) + vwsp(i,k);$
    $wdir(i,k) = f_{wdir}(wdir(i,k-1)) + vwdir(i,k);$
    $wind(i,k) = [wsp(i,k), wdir(i,k)];$
    Randomly generate $w(i,k)$ according to normal
    distribution $N(0, {\sigma_w}^2)$;
    $B(i,k) = calA(PF(wind(i,k))) + w(i,k);$
    $weights(i,k) = obA(k+1) - obA(k) - B(i,k);$
    $weights(i,k) = \frac{1}{\sigma_w \sqrt{2\pi}} e^{-\frac{weights(i,k)^2}{2\sigma_w^2}};$
3. Normalize weights
$s\_wts = 0;$
for $i = 0$ to $N-1$
    $s\_wts = s\_wts + weights(i,k);$
for $i = 0$ to $N-1$
    $n\_wts(i,k) = weights(i,k) / s\_wts;$
4. Resampling
$q(0) = n\_wts(0,k);$
for $i = 1$ to $N-1$
    $q(i) = q(i-1) + n\_wts(i,k);$
Uniformly generate $N$ numbers between 0 and 1,
and sort them as array $u$;
$count = 1;$
for $j = 0$ to $N-1$
    while $(q(count) < u(j))$
        $count = count + 1;$
    $temp(j) = wind(count,k);$
for $l = 0$ to $N-1$
    $wind(l,k) = temp(l);$
5. Output states (the estimated wind data)
$os(k) = 0;$
for $i = 0$ to $N-1$
$os(k) = os(k) + wind(i,k) * n\_wts(i,k);$
```

In this algorithm, step 1 initializes N particles. With time advances, step 2 to step 5 are executed as shown in Figure 2.

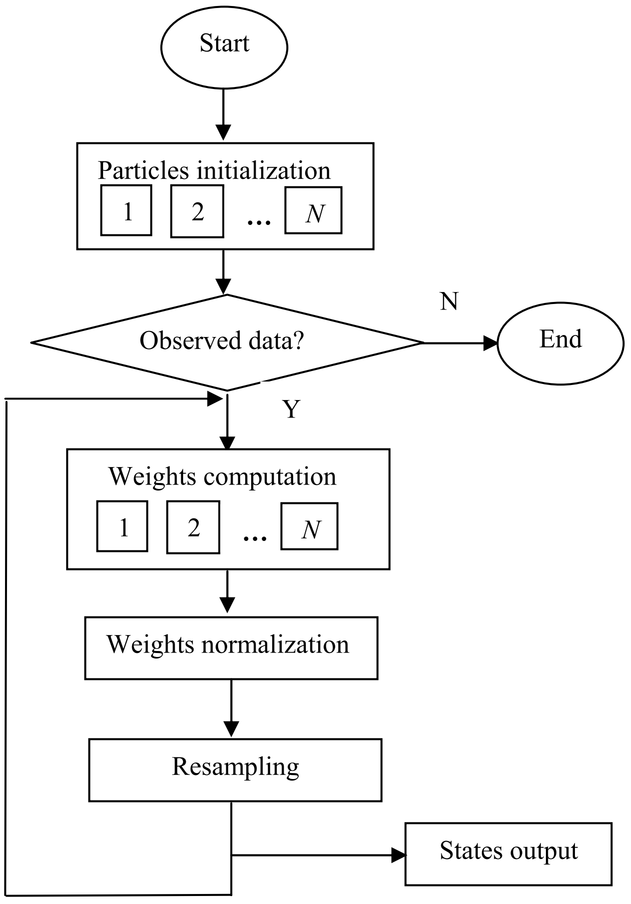

Figure 2: Flow of particle filters algorithms of case study

There are several proposed sampling algorithms, for example, systematic sampling (Kitagawa 1996) and residual sampling (Liu et al. 1998). Among them, the systematic sampling is widely used (Hol 2004). We use systematic sampling in our implementations too.

6 EXPERIMENTS AND RESULTS

6.1 Experiments Design

Based on the example described above, we design two experiments as follows. The first one uses a uniform fuel model (fuel model 7) with simple wind flow, and zero slope and aspect. The wind is generated as follows. The initial wind speed is 2 miles/hour. In the first half of the entire process that lasts for 5 hours, the wind speed increases 0.5 mile/hour every time step (10 minutes), and then decreases 0.5 mile/hour every time step. The wind direction keeps a fixed value of 180 degrees. In the second experiment, we use non-uniform GIS data, where cells have different fuel models, aspects, and slopes. The initial wind speed and wind direction are 5 miles/hour and 180 degrees. Then every time step, the wind speed increases or decreases a number between 0~2 miles/hour based on that of the last step, and the wind direction is 180±20 degrees.

Table 1 and Table 2 show the real wind data (column 2 and column 3) generated based on the wind flow model described above. Note that these are the data that need to be estimated by the particle filters. Based on the real wind data, we use DEVS-FIRE to run the simulations to obtain the fire fronts and their corresponding burned areas every 10 minutes (these are the observation data). Table 1 and Table 2 also show the weather data (column 4 and 5). We assume the weather data are known only at every 1 hour. Our goal is to estimate the wind data every 10 minutes for the time when the weather data are not available. This is achieved using particle filters based on the observation data of fire fronts and burned areas. The particle filters algorithm uses 50 particles. The simulations are run for 5 hours, and the wind flow model used in the particle filters for both experiments is as follows.

$$\begin{cases} wsp(t) = wsp(t-1) + N(0,4), t = 1,2,... \\ wdir(t) = 180 + N(0,40), t = 1,2,... \end{cases}$$

Table 1: Wind data of experiment 1

Time	Real wsp	Real wdir	Weather wsp	Weather wdir
0:00	2.0	180	2.0	180
0:10	2.5	180	2.0	180
0:20	3.0	180	2.0	180
0:30	3.5	180	2.0	180
0:40	4.0	180	2.0	180
0:50	4.5	180	2.0	180
1:00	5.0	180	5.0	180
1:10	5.5	180	5.0	180
1:20	6.0	180	5.0	180
1:30	6.5	180	5.0	180
1:40	7.0	180	5.0	180
1:50	7.5	180	5.0	180
2:00	8.0	180	8.0	180
2:10	8.5	180	8.0	180
2:20	9.0	180	8.0	180
2:30	9.5	180	8.0	180
2:40	9.0	180	8.0	180
2:50	8.5	180	8.0	180
3:00	8.0	180	8.0	180
3:10	7.5	180	8.0	180
3:20	7.0	180	8.0	180
3:30	6.5	180	8.0	180
3:40	6.0	180	8.0	180
3:50	5.5	180	8.0	180
4:00	5.0	180	5.0	180
4:10	4.5	180	5.0	180
4:20	4.0	180	5.0	180
4:30	3.5	180	5.0	180
4:40	3.0	180	5.0	180
4:50	2.5	180	5.0	180

Table 2: Wind data of experiment 2

Time	Real wsp	Real wdir	Weather wsp	Weather wdir
0:00	6.0	186	6.0	186
0:10	5.2	178	6.0	186
0:20	6.1	199	6.0	186
0:30	6.9	174	6.0	186
0:40	5.4	187	6.0	186
0:50	6.1	191	6.0	186
1:00	7.1	182	7.1	182
1:10	7.2	166	7.1	182
1:20	7.0	169	7.1	182
1:30	8.3	167	7.1	182
1:40	7.8	172	7.1	182
1:50	8.9	182	7.1	182
2:00	10.4	182	10.4	182
2:10	10.7	191	10.4	182
2:20	12.2	179	10.4	182
2:30	10.5	179	10.4	182
2:40	9.9	173	10.4	182
2:50	11.6	185	10.4	182
3:00	10.9	169	10.9	169
3:10	11.8	176	10.9	169
3:20	12.6	187	10.9	169
3:30	10.9	189	10.9	169
3:40	12.4	168	10.9	169
3:50	11.4	177	10.9	169
4:00	10.8	196	10.8	196

4:10	11.1	180	10.8	196
4:20	9.3	166	10.8	196
4:30	7.8	186	10.8	196
4:40	6.5	181	10.8	196
4:50	5.5	185	10.8	196

6.2 Experiments Results

6.2.1 Case 1: Uniform Fuel with Simple Wind Condition

Figure 3 displays the states of wind speed every 10 minutes for case 1. Figure 4 shows the states of wind direction of each time step of case 1. In the figures, real data mean wind data used to obtain observed areas in particle filters, and weather data, estimated data stand for the wind data every 1 hour and estimated wind data by particle filters respectively. From Figure 3 and Figure 4, we can see that

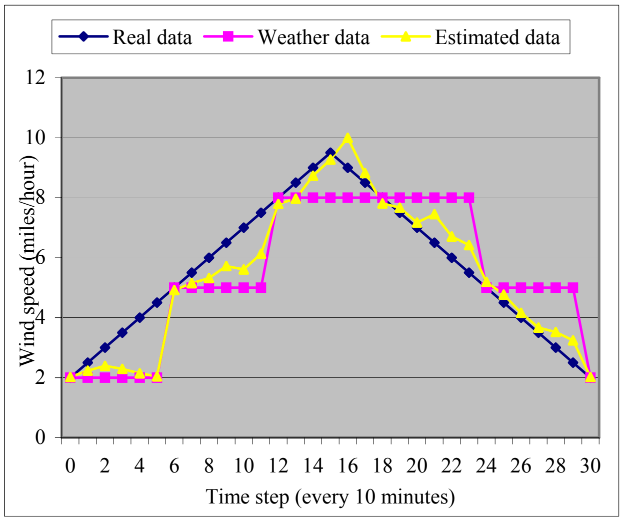

Figure 3: Wind speeds of case 1

the estimated wind speeds and wind directions have the same trend as those of the real wind conditions. Therefore, in the practical applications, if we only know the wind condition every time period (e.g., 1 hour), we can estimate the wind data each time slot between this time according to observed data by particle filters. Figure 5 displays the burned areas with three wind conditions form time step 11 to 20. From the figure, we can conclude that compared to using weather data, the burned areas computed by using estimated wind data are closer to the ones calculated by real wind data. This means that the estimated data can be used to produce more accurate predictions.

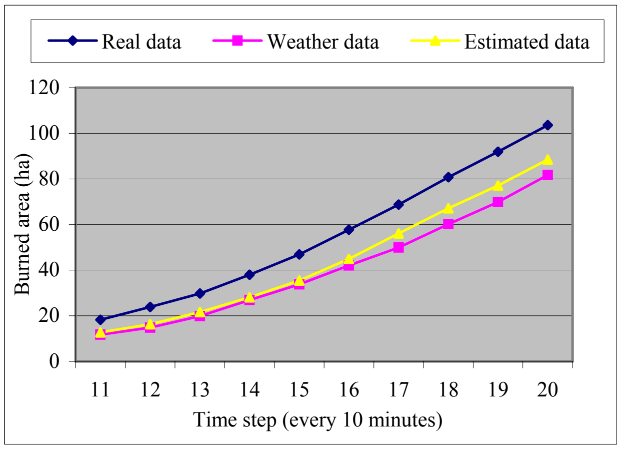

Figure 5: Burned areas of case 1

6.2.2 Case 2: GIS Data with Complex Wind Condition

Figure 6 and Figure 7 show the wind speeds and wind directions of case 2. From the pictures, we can draw similar conclusions as before. Figure 8 displays the burned areas with the three types of wind data from time step 21 to 24. From the picture, we can see that the estimated areas fall between the areas obtained by using real data and by using weather data.

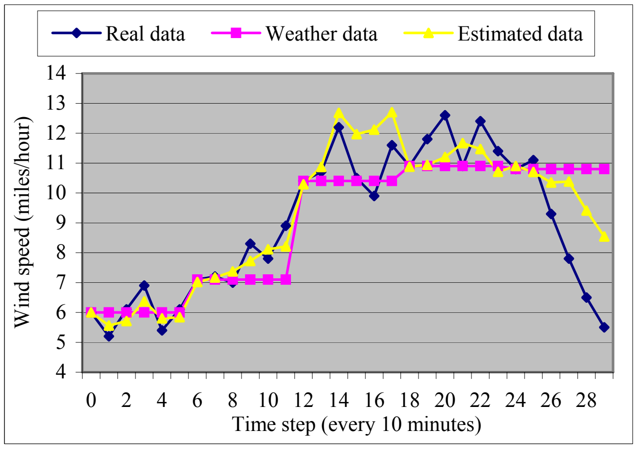

Figure 6: Wind speeds of case 2

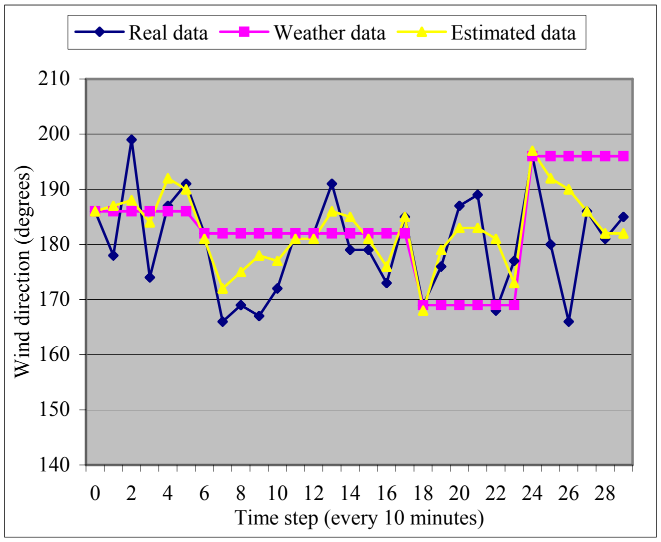

Figure 7: Wind directions of case 2

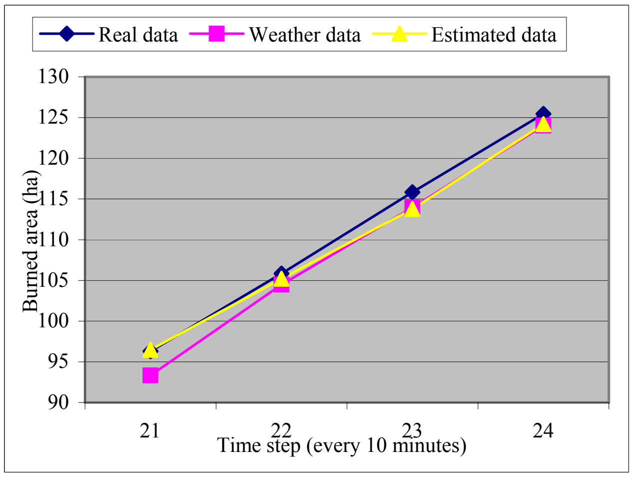

Figure 8: Burned areas of case 2

7 CONCLUSIONS

In this paper, we discussed the basic knowledge of particle filters method, and then constructed the non-linear dynamic system model of DEVS-FIRE fire spread model according to the specification of sequential Monte Carlo methods. Although there are many possible applications of particle filters using DEVS-FIRE, we focus on the wind estimation as a case study in the paper. Preliminary results show that particle filters are powerful tools that can be used in the problem of wildfire spread simulation. Future work will proceed along the following directions: 1) precisely estimate the fire shapes by using observation data from satellite images; 2) apply particle filters to estimate the static parameters from the observation data, and smooth parameters to study past states according to observation data; 3) develop advanced methods to reduce the computation cost of particle filters that use large number of particles.

REFERENCES

Andrews, P. L., C. D. Bevins, R. C. Seli. 2005. BehavePlus fire modeling system, version 3.0: User's Guide Gen. Tech. Rep. RMRS-GTR-106WWW Revised. Ogden, UT: Department of Agriculture, Forest Service, Rocky Mountain Research Station, 132 p.

Bølviken, E., P. J. Acklam, N. Christophersen, and J. -M. Størdal. 2001. Monte Carlo Filters for Non-linear State Estimation. *Automatica* 37(2):177–183.

Bradley, Justin M. 2007. Particle Filter Based Mosaicking for Forest Fire Tracking. Master thesis, Brigham Young University.

Chib, S. and E. Greenberg. 1995. Understanding The Metropolis–Hastings Algorithm. *The American Statistician* 49(4):327–335.

Coen, J. L., J. Mandel, et al. 2007. A Wildland Fire Dynamic Data-driven Application System. *11th Symposium on Integrated Observing and Assimilation Systems for the Atmosphere, Oceans, and Land Surface* (*IOAS-AOLS*).

Doucet, A., S. J. Godsill, and C. Andrieu. 2000. On Sequential Monte Carlo Sampling Methods for Bayesian Filtering. *Statistics and Computing*, 10(3):197–208.

Doucet, A., N. D. Freitas, N. Gordon (eds.). 2001. *Sequential Monte Carlo methods in practice*. New York: Springer-Verlag.

Doucet, A.,V. Tadic. 2003. Parameter Estimation in General State-space Models Using Particle Methods, *Ann. Inst. Statist. Math.* 55(2): 409-422.

Douglas, C. C., D. B . Jonathan, et al. 2006. DDDAS Approaches to Wildland Fire, Modeling and Contaminant Tracking. *Proceedings of the 2006 Winter Simulation Conference*, 2117-2124.

Finney, Mark A. 1998. FARSITE: Fire Area Simulator–Model Development and Evaluation, United States Department of Agriculture Forest Service Rocky Mountain Research Station Research Paper, RMRS-RP-4 Revised March 1998, revised February 2004.

Fox D., Thrun S., Burgard W. & Dellaert F. 2001. Particle Filters for Mobile Robot Localization. In *Sequential Monte Carlo Methods in Practice* (eds A. Doucet, J. F. G. de Freitas and N. J. Gordon). New York: Springer-Verlag.

Gibson, S. H. 2003. Maximum Likelihood Estimation of Multivariable Dynamic Systems via the EM Algorithm. PhD thesis, The University of Newcastle, Newcastle, NSW, Australia.

Gordon, N. J., D. J. Salmond, and A. F. M. Smith. 1993. Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation. In *IEE Proceedings on Radar and Signal Processing* 140: 107–113.

Gustafsson, F., F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and R. -J. Nordlund. 2002. Particle Filters for Positioning, Navigation and Tracking. *IEEE Transactions on Signal Processing* 50(2):425–437.

Hol, J. 2004. Resampling in Particle Filters. Master Thesis, Department of Electrical Engineering, Linköping University, Sweden.

Jazwinski, A. H. 1970. Stochastic Processes and Filtering Theory. *Mathematics in science and engineering*. Academic Press, New York, USA.

Kailath, T., A. H. Sayed, and B. Hassibi. 2000. Linear Estimation. *Information and System Sciences Series*. Prentice Hall, Upper Saddle River, NJ, USA.

Kitagawa, G. 1996. Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models. *Journal of Computational and Graphical Statistics* 5(1):1–25.

Liu, J. S. and E. Chen. 1998. Sequential Monte Carlo Methods for Dynamics Systems. *Journal of American Statistical Association* 93:1032–1044.

Mandel, J., D. Jonathan, et al. 2007. A Wildland Fire Model with Data Assimilation. CCM Report 233, *Mathematics and Computers in Simulation*.

Natimo, L., X. Hu, and Y. Sun. 2008. DEVS-FIRE: Towards an Integrated Simulation Environment for Surface Wildfire Spread and Containment, *SIMULATION: Transactions of The Society for Modeling and Simulation International*, Vol. 84, Issue 4, April 2008: 137-155.

Schön, Thomas B. 2006. Estimation of Nonlinear Dynamic Systems– Theory and Applications. Ph.D. Dissertation, Linköpings University, Sweden.

Simandl, M., Straka, O. 2007. Sampling Densities of Particle Filter: A Survey and Comparison. *American Control Conference,* 4437-4442.

Wang, X. D., Rong Chen, Jun. S. Liu. 2002. Monte Carlo Bayesian Signal Processing for Wireless Communications, *Journal of VLSI Signal Processing Systems*, 30(2002)1-3, 89-105.

Zhang, J., Rong Chen, C. Tang, J. Liang. 2003. Origin of Scaling Behavior of Protein Packing Density: A Sequential Monte Carlo Study of Compact Long Chain Polymers. *JOURNAL OF CHEMICAL PHYSICS*, 118(13): 5102-6109.

Zeigler, B. P., H. Praehofer, and T. G. Kim. 2000. *Theory of modeling and simulation*, 2nd Edition, Academic Press.

AUTHOR BIOGRAPHIES

FENG GU is a Ph.D. candidate in the Computer Science Department at Georgia State University. His research interests include modeling and simulation and system validation and calibration.

XIAOLIN HU is an assistant professor in the Computer Science Department at Georgia State University. His research interests include modeling and simulation, autonomous agent and multi-agent systems, and simulation-based development.