SpSIR: A Spatially-Dependent Sequential Importance Resampling For High Dimensional Spatial Temporal System Simulation

Yuan Long and Xiaolin Hu
Computer Science Department
Georgia State University
Atlanta, GA 30303, USA
ylong4@student.gsu.edu and xhu@gsu.edu

ABSTRACT
For a high dimensional spatial temporal system, data assimilation is introduced to dynamically adjust the simulation result when the model is imperfect or the parameters are imprecise. Sequential Monte Carlo (SMC) methods or particle filters (PFs) are popular data assimilation scheme. However, in a high dimensional spatial temporal system where observations and state are spatially distributed, directly applying SMC method may lead to poor prediction result or high computational cost because of the sample size. In this paper, we break a full state and observations into spatial regions and propose a new resampling algorithm: SpSIR. This algorithm exploits the spatial locality property and employs a divide and conquer strategy to reduce state dimension and data complexity. A case study in wildfire simulation demonstrates that the proposed SpSIR algorithm improves the performance of prediction event when sample size is not quite large.

Author Keywords
Particle filter; high dimensional; data assimilation; resampling.

ACM Classification Keywords
I.6.8 [SIMULATION AND MODELING]: Types of Simulation – Monte Carlo

INTRODUCTION
In a high dimensional spatial temporal system, simulation model is always very complex and the number of parameters is quite large. An imperfect model or imprecise parameters may introduce errors to the simulation result. Therefore we need to assimilate real observation data into the system and dynamically adjust the prediction result.

In recent years, PFs have become a very popular tool in ensemble data assimilation to model and simulate different spatial temporal systems, such as wildfire simulation, traffic simulation and target tracking [1-4].

PFs are sample-based methods, which represents the prediction of state by a posterior distribution of samples [5].

In a spatial temporal system, the samples are drawn from space of unknowns and filtered by observation. However, applying PFs into a high dimensional system has two main challenges. A major challenge is due to large number of state variables. As the state space is very large, small number of particles is difficult to achieve satisfactory results by having "correct combination" of all state variables. Another challenge is from observation data. In standard particle filter, the importance weights of different particles for any chosen state variable are influenced by all observation data, even if those observation data are nearly independent of the particular state variable [6, 7].

For a high dimensional spatial temporal system whose states and observation data are spatially distributed and have finite correlation lengths, the standard particle filters thus overestimate the information available in the observation data and underestimate the uncertainty of the posterior distribution. Consider wildfire as an example, the observation data (e.g., ground temperature sensor data) from different regions of the fire typically reflect only the fire states in their corresponding regions, not others. Motivated by the spatial locality property of both system state and observation data, we extend the standard bootstrap filter algorithm and propose a spatially-dependent sequential importance resampling (SpSIR) algorithm. This algorithm breaks state and observation into spatial regions and employs a divide and conquer strategy to reduce state dimension and data complexity.

The rest of paper is organized as follows. First we introduce the recent solutions for high dimensional spatial temporal system modeling and simulation. Next, an overview of standard particle filter (the bootstrap filter) is provided. After that, we describe the global framework and detailed steps for SpSIR algorithm. Then a case study in wildfire simulation demonstrates the improvement of SpSIR algorithm. Finally, we draw a conclusion and point out the future work.

RELATED WORK
Although particle filters based data assimilation method has been proved as an improvement for high dimensional spatial temporal system simulation, the dimensionality is
still a curse of particle filtering and thus impacts the performance of simulation [8, 9]. Recently, it has gained a lot attention from the researchers. Different strategies are studied to improve the accuracy and reduce the required number of samples. State partition is one of the strategies that improve performance of data assimilation method.

State partition is an approach that divides state into multiple partitions and then applies particle filters. Partition sampling proposed in [10] is the first method using state partition concept in multiple object tracking. Especially, each object is a single partition. Besides, the dynamics and resampling operations are sequentially applied in a hierarchical scheme. Later, MacCormick et al. implemented this method in hand tracking and again proved the advantage of partition sampling [11]. Also for the hand tracking application, Brandao et al. presented a similar subspace hierarchical filter, which uses the implicit structure of tracked object to partition the state space [12]. Besides the object tracking in computer vision, the state partition is also used in other applications. For example, in multiple targets tracking, where the sensors are more diverse, Djuric et al. partitioned the state dimensions into subspaces and applied particle filter in each partition for target [4]. Particularly, the state is decomposed according to the measurements from variance of sensors (e.g. signal strength, angle of signal arrive, direction of motion and absolute velocity of target).

This paper will use “state partition” to improve the prediction performance in high dimensional spatial temporal system. Overall, the recent state partition methods have been demonstrated as an improvement of particle filter for high dimensional spatial system. However, because of the diversity of implementation for variance of applications, new problems emerge when applying those methods to a new application. Two major problems are observation data association problems (described in introduction) and sampling problem. Although, some state partition methods use sequential sampling for the partitions, in some cases, the sequential sampling cannot be implemented by model, such as in cell-based wild fire simulation, where the state of each cell depends on all neighboring cells in a 2D space.

OVERVIEW OF PARTICLE FILTER METHODS
A spatial temporal system is a dynamic system, represented by two equations. One is the state transition equation; the other is the likelihood equation for measurements. The two equations are as follows:

\[ x_t = f(x_{t-1}, v_t) \]  
\[ y_t = g(x_t, A_t) \]

Where transition function \( f \) shows how the current state \( x_t \) evolves from previous state \( x_{t-1} \) and measurement function \( g \) calculates the current observations \( y_t \) based on predicted state \( x_t \). \( v_t \) and \( A_t \) are two independent variables representing noises to transition and measurement function respectively.

To simulate a spatial temporal system, a PFs based dynamic data driven data assimilation method has been introduced in [2]. The basic framework of this method is built on the bootstrap filter (a sequentially importance resampling method), which contains three main steps at each iteration: state transition, weight calculation and resampling steps. Let \( x_t^i \) be the state represented for \( i \)th particle where \( i = 1, 2, \ldots, N \) (\( N \) is the sample size). At every time step, based on previous samples \( x_{t-1} \), we draw new \( N \) samples \( x_t \) by using transition function \( f \). Then at weight calculation step, measurement function \( g \) maps each new sample with simulated observations. After that, by comparing simulated observation with real observation \( y_t \), each sample is weighted through likelihood function in equation (3). Next, all the samples are resampled according to the normalized weight \( w(x_t^i) \) calculated by equation (4).

\[ \varphi(x_t^i) = w(x_{t-1}^i)p(y_t|x_t^i) \]  
\[ w(x_t^i) = \frac{\varphi(x_t^i)}{\sum_{i=1}^{N} \varphi(x_t^i)} \]

SPATIALLY-DEPENDENT SEQUENTIAL IMPORTANCE RESAMPLING (SPSIR)

We extend the standard bootstrap filter and propose a new SPSIR algorithm. The basic idea of SPSIR algorithm is to break system state and observation data into different spatial regions and explicitly set an observation’s influence to its spatially local neighborhood according to the spatial region when computing particle’s weight.

Proposed method
Let \( r = \{r_1, \ldots, r_m\} \) be the entire space (i.e. a real map), which is divided into \( m \) smaller regions \( r_j \) for \( j = 1 \ldots m \). Since both the state and observation data are spatially dependent, they can be broken into subgroups according to the divided regions.

There are many approaches to break a space, such as differentiating the cities, towns, lakes and forests in a real map, clustering observer’s observation area and gridding the map into some small rectangles. As map gridding needs the fewest information of a system space and is straightforward, in this paper, we will use that for state partition in case study of wildfire simulation.

Since we already known how to divide space and get substates in corresponding regions, the improved framework for SPSIR algorithm is designed as shown in figure 1. Similar to standard bootstrap filter, it contains the three main steps.

1) Integrated sampling: The sampling step shares the same sampling method in the bootstrap filter. In this step, based on the last step’s state (a full state \( x_t \)), current state is calculated according to the state equation function (i.e. state transition model).

971
2) **Weight calculation**: It differs significantly from the bootstrap filter algorithm. Instead of assigning all observations to a full state, SpSIR breaks each full state $x_t^j$ (from set $x_t$) into multiple sub-states $\{x_{t,r_1}^i, ..., x_{t,r_m}^i\}$ according to divided region $\{r_1, ..., r_m\}$ and associate each sub-state with validated real observations $\{y_{t,r_1}^i, ..., y_{t,r_m}^i\}$. Then update the weight $\omega_{r_j}^i$ for region $r_j$ according to associated observations $y_{t,r_j}^i$ and simulated observations calculated by measurement function. At last, for each region, we calculate the normalized weight $w_{r_j}^i$ for sub-state $x_{t,r_j}^i$.

3) **Resampling**: Instead of resampling a full state, we resample the sub-states set $\{x_{t,r_1}^i, ..., x_{t,r_m}^i\}$ according to the weight set $\{w_{t,r_1}^i, ..., w_{t,r_m}^i\}$ for each region $r_j$. Besides the standard resampling, sub-states with same indexes from different regions in sub-states set $\{x_{t,r_1}^i, ..., x_{t,r_m}^i\}$ are combined to form a full state.

In the following sections, these three steps will be described in details.

**Integrated Sampling**

This step shares the same sampling method as in bootstrap particle filter. By applying the transition function $f(x_{t-1}, v_t)$, the full state $x_{t-1}$ in each particle is evolved to state $x_t^i$ for current step.

However, at this time step, the inputs for $i$th particle are the combined partitioned sub-states $x_{t,r_1}^i, ..., x_{t,r_m}^i$ generated by resampling at previous time step. We have to combine those sub-states before the state transition. This is because in some applications, the transition models need a full state to accomplish sampling step, such as DEVS-FIRE [13,14] for wildfire simulation.

**Weight Calculation**

At weight calculation step, each particle should be assigned a weight $w^i$ according to the likelihood function $p(y_t^i | x_t^i)$. But unlike the bootstrap particle filter, in SpSIR algorithm, every particle needs a set of weights $\{\omega_{r_1}^i, ..., \omega_{r_m}^i\}$ for corresponding sub-states set $\{x_{t,r_1}^i, ..., x_{t,r_m}^i\}$ instead of one weight for a full state. The weight calculation for substate $x_{t,r_j}^i$ is like below:

$$
\omega_{r_j}^i = p(y_t^i | x_{t,r_j}^i)
$$

In equation (5), the likelihood function for each sub-state is based on the impacted observations $y_{t,r_j}^i$, but not all the real observations. This owes to sensor's locality characteristic in a large spatial temporal system. The locality implies that the widely distributed sensors only reflect the state situation limited in its own small observation area. Therefore, one region area is observed by certain number of sensors but maybe not all sensors. Also, observations from current sub-state may reflect information from other sub-states. Consider the wildfire as an example. In figure 2, the state space is decomposed into 4 regions and red line shows the fire front. Each temperature sensor has an observation range represented by gray circle. Sensor A’s observation range limits in region 1 and only reflects the state in region 1; Similarly, sensor B’s observation range limits in region 4 and only reflects the state in region 4. However, sensor’s observation area can also across one or more sub-states like sensor C and sensor D. Although sensor C locates at region 2, the reading of it is only impacted by the fire spread information from region 1 in this example. Sensor D’s observation range across both region 1 and region 3 and could reflect fire spread information for both region 1 and region 3. But assigning Sensor D to which region or substate is a problem.
Figure 2. Example of sensor’s locality characteristic in wildfire simulation.

If sensor's observation area crosses one or more sub-states, we need to assign the real observation to the most possible sub-states. Then it forms a classical data association problem. In a real environment, the problem may be very complex because of the sensor's nature (e.g. temperature sensor, camera). So we need to consider the following two situations to solve the data association problem in a high dimensional spatial temporal system.

1. Sensor can reflect information for only a single substate

Some kinds of sensors’ measurement are impacted by a particular point. For example, the measurement from temperature sensor is impacted by the highest temperature point nearing it (e.g. the closest ignition point in wildfire). This particular point is located in only one region. It indicates that this sensor reflects the information for only one substate.

To associate observation data from this kind of sensor to one substate, many methods such as suboptimal nearest neighbor (SNN) and global nearest neighbor (GNN) [15] have been developed in target tracking problem (in which each target could be considered as a moveable substate). In this paper, we use the following algorithm to find the possible substate in a more general large temporal spatial system where the substate is constrained in a static region.

**Step1:** For each sensor in the fth particle, search sub-state within its observation range. If substate not found, discard this sensor and assign it to none of the sub-states; if found, go to step2.

**Step2:** Apply the same manner as in bootstrap particle filer and find the most possible impacted points in every substate.

**Step3:** Compare all those impacted points, then select the most possible point and record the substate it belongs to.

**Step4:** Assign the observation data to the recorded substate.

2. Sensor can reflect information for multiple sub-states

Some kinds of sensors' reading are impacted by a line or area. For example, camera sensor records a two dimensional image for an area. Also, in the traffic simulation, the density sensor reflects the density in a road segment (as shown in figure 3). If the area or line segment falls in different sub-states, we need to find a solution to separate their impact to related sub-states. There are two methods to solve this problem.

A. Separate sensor's observation value according to the ratio of observed regions' length or area and use the separated observation value as the real observation to obtain final weight.

B. Define a probability density function dynamically according to the observation data and apply this density function to each substate to compute final weight.

Take the traffic simulation for example in the following figure 3, where a sensor located (displayed by a dash rectangle box) in the road segment detects the number of cars on road. The sensor’s observation range is also limited in the dash rectangle box. And the state space is divided into two regions according to the vertical dash line. This sensor observes 6 vehicles in its observation range (crossing both right and left regions). One of the particles shows 4 vehicles in left region, and 6 vehicles in right region as illustrated in figure 3. For that sensor, the simulated observation data is 2 vehicles in left observed region and 5 vehicles in right observed region.

![Figure 3. Example of traffic simulation.](image)

To associate real observation to the substate, if applying method A, as the road lengths for the two observed regions has a ratio as 1:2, we need to separate the real observation into two parts: 2 for right and 4 for left. After that, we can use Gaussian distribution as a probability function to calculate the weight.

If applying method B, for each region, the value for number of vehicles between 1 and 6 has a higher probability than other values. Then we can design a probability density function where probability for number 1 to 6 are equal and high but decreases when number increases from 6. After that, the weight is calculated based on this function for each substate. This probability function can also be considered as likelihood function in this example.

**Resampling**

After the weight calculation step, for each region \( r_j \), we resample from each set of sub-state \( \{x^1_{r_j}, \ldots, x^N_{r_j}\} \) according to their normalized weight \( w^1_{r_j} \ldots, w^N_{r_j} \). The resampled sub-state set for region \( r_j \) is \( \{x^1_{r_j}, \ldots, x^N_{r_j}\} \).

The resampled sub-states are grouped together to form a full state through randomly generated particle index. Firstly for ith particle we randomly generate m numbers \( k_1 \) to \( k_m \).
according to the uniform distribution $U(N)$. Secondly sub-states $x_{t+1,r_j}^{k_1}$ to $x_{t+1,r_j}^{k_m}$ form a group for full state $x_{t+1,r_j}^t$.

The partitioned sub-states in each group are integrated into a full state $x_t^t$ to accomplish sampling step. In this paper, we assume the sub-states are not so highly dependent that sub-states are spatially combined and integrated into a full state $x_t^t$.

**CASE STUDY IN WILDFIRE SIMULATION**

Wildfire simulation is a typical high dimensional spatial temporal system, where the state searching space is as large as $M^2$ for a $M \times M$ cell space. In the experiment, we used wildfire simulation as a case study. The state was the fire front and observation data were from ground temperature sensors. The transition function was based on DEVS-FIRE as described in [13, 14]; the measurement function was based on a heat model in [2].

**Experiment Settings**

Because it is hard to get the real observation data and real state information from physical environment, we chose to use identical-twin experiment to evaluate the effectiveness and performance of employed data assimilation method. The detail of identical twin can be found in [2]. For correct weather information, wind speed was 8 (mph) and wind direction was 180 (degrees) with random variances added every 30 minutes. Similarly, for the “erroneous” weather information, wind speed was as 10 (mph) and wind direction was as 180 (degrees) with random variances added every 30 minutes. The random variance ranged from 2 to -2. The sensors were deployed every 10 cells in a 200x200 cell space. And observation range radius for each sensor was 30.

In order to show the improvement of SpSIR algorithm, we mainly divided the cell space into 4 and 7 regions and run experiments with particle number 100 respectively. Overall, four cases were designed. Case 1: the searching cell space was not divided. Case 2: the searching cell space was divided into 4 equal regions. Case 3: initially the searching cell space was divided into 4 equal regions, and then one of them was divided into 4 regions again. Case 4: the searching space was not divided, but the particle number was increased to 200. Each case uses 8 time steps in the experiment. And the duration for each time step was 20 minutes. The original ignition point was at the center of cell space.

**Experiment result and analysis**

The fire fronts at last time step for all 4 cases are displayed in figure 4.
In each case, filtered fire front was compared with real fire front and simulated fire front. The filtered fire front (drawn in red line) was generated with erroneous weather information and produced by the SpSIR algorithm described in this paper; real fire front (drawn in black line) was generated by pure DEV5-FIRE with real weather information; simulated fire front (drawn in blue line) was generated by pure DEV5-FIRE with erroneous weather information. In case 1 and case 4, the whole space was considered as one region. In case 2, the four regions were indexed from R1 to R4. In case 3, region R4 in case 3 were further partitioned into 4 regions indexed from R5 to R8 and seven regions were considered. The filtered fire fronts for case 2 and case 3 were the combination of best particles in each region; for case 1 and case 4, the filtered fire fronts were the best particles for the single region.

In all 4 cases, as the wind speed for filtered fire and simulated fire was faster than in real fire, the fire fronts of them tended to spread more widely than in real fire as shown in figure 4.

One strategy to improve the accuracy is to partition the state by using SpSIR algorithm. Because the ignition point was at center, we found that the real fire spreads fast in R4 and spread slowly in R1, R2 and R3.

As fire spread fast in region R4, larger searching space was needed for region R4 to achieve satisfactory result. So from figure 4, we can see that by using multiple sub-states, the filtered fire was closer to the real fire. In case 1, the fire front deflected obviously from the real fire in region R4. But, this deflection was “pulled back” by applying 4 sub-states in case 2. It was even further reduced by applying 7 sub-states in case 3. This is because the filtered fire front approached the real fire front in region R7 and R8.

As fire spread slowly in region R1, R2 and R3, searching space was comparative smaller and multiple sub-states still generated better result in most regions. But in region 1 for both case 2 and case 3, there was an unexpected fire head neighboring to region 2 in the filtered fire. This happened because some sensors located at nearby border had been assigned to region 2 and had no contribution to weigh the state in the area (region 1) containing that unexpected fire head. Thus even the sub-state in region 1 had the wrong fire head, it was still assigned as a higher weight. Although, the filtered front in those regions had no obvious closer to the real fire front for case 2 and case 3, the accuracy still outperformed the result when using more particles.

Another traditional strategy to improve the accuracy is increasing the particle number. Comparing case 1 with case 4, we found that the whole fire front was closer to the real fire as more particles were used to explore the searching space. Especially in region 4, the estimation was comparable with the result by using 7 sub-states. However, as the weight was calculated based on a whole state, in order to search proper states for R4, the originally good result in region R1, R2 and R3 were sacrificed and even much worse than using less particles.

Figure 5 shows the symmetric differences (i.e. the number of cells with different status) compared with real fire for all time steps. Although, the symmetric differences for case 3 was even greater than that in case 2 in the first several steps, but at the final time step, case 3 had the minimum symmetric differences. This unexpected trend happened because before step 6, only R5 of the four sub regions in R4 contained ignition points, the other regions R6 to R8 were empty and had no contribution to the corresponding sub-states. However, fire eventually spread to the other three regions at following steps and sub-states in them were weighted and resampled to achieve better results finally.

![Comparison of Symmetric Differences](image)

**Figure 5. Comparison of symmetric differences among all four cases.**

**CONCLUSION**

In this paper, we present a new resampling algorithm SpSIR. This algorithm could be used to assimilate observation data into high dimensional spatial temporal system simulation and thus reduce the computation cost and improve the prediction accuracy than traditional PFs based data assimilation framework. SpSIR algorithm employs the divide and conquer methodology. It divides the sample space into multiple regions, one state then partitioned into multiple sub-states and observations are decomposed into sub-groups for sub-states through data association algorithm. After that, we calculate weights for each sub-state respectively. The weighted sub-states form a sub-group according to the region and are resampled. By applying the state partition idea, spatially good part of particle has a higher chance to be chosen and as a result improve the whole particle’s accuracy by combination. The case study in wildfire simulation shows this algorithm improves prediction accuracy significantly even when sample size is not large enough to cover the whole state space and achieve satisfactory result in standard particle filters. Our future work is to apply SpSIR algorithm into traffic simulation, where the data association problem will be more complicated.
REFERENCES