Data Assimilation in Discrete Event Simulations – A Rollback based Sequential Monte Carlo Approach

Xu Xie
Delft University of Technology
Delft, The Netherlands
x.xie@tudelft.nl

Alexander Verbraeck
Delft University of Technology
Delft, The Netherlands
a.verbraeck@tudelft.nl

Feng Gu
College of Staten Island, CUNY
2800 Victory Blvd., Staten Island, NY 10314
Feng.Gu@csi.cuny.edu

ABSTRACT
Data assimilation is an analysis technique which aims to incorporate measured observations into a dynamic system model in order to produce accurate estimates of the current state variables of the system. Although data assimilation is conventionally applied in continuous system models, it is also a desired ability for its discrete event counterpart. However, data assimilation has not been well studied in discrete event simulations yet. This paper researches data assimilation problems in discrete event simulations, and proposes a novel and efficient data assimilation algorithm – the rollback based Sequential Monte Carlo (SMC) method. The proposed algorithm employs only one simulation to generate particles by exploiting the rollback technique, therefore, it consumes much less memory compared with the standard SMC methods. To evaluate the accuracy of the rollback based SMC method, an identical-twin experiment in a discrete event traffic case is carried out and the results are presented and analyzed.

Author Keywords
Data Assimilation, Discrete event simulations, Sequential Monte Carlo methods, Rollback.

1. INTRODUCTION

Computer simulations have long been used for studying and predicting the behavior of complex systems [14]. However, accurate analysis and prediction of the behavior of complex systems are difficult, because even complex models are still lacking the ability to accurately describe such systems [7], therefore, even elaborate complex models of systems produce simulations that diverge from or fail to predict the real behavior of those systems. This situation is accentuated in cases where real-time dynamic conditions exist [7].

The availability of real-time observations from real systems has increased along with the advances in sensor technology. The increased availability of data allows for a new simulation paradigm – *dynamic data driven simulation*, where a simulation is continually influenced by the real time data streams for better analysis and prediction of a system under study [14].

The core technique in dynamic data driven simulation is *data assimilation*, in which observations are incorporated into a dynamic system model to produce accurate estimates of current states of the system [21].

Data assimilation has been applied with success in many applications, such as weather forecasting [15], chemical data assimilation [6], ocean data assimilation [4], etc. But in these applications, systems are continuous and are conventionally modeled as (*partial*) *differential equations*, and these differential equations are again computed using *numerical methods*, and thus approximated by *difference equations* [24]. Besides continuous systems and models, a large number of discrete event systems and models exist in practice, such as manufacturing systems, queuing networks, etc. In discrete event systems, entities are usually represented with discrete state variables which evolve at discrete moments over continuous time, and change their values due to the occurrence of particular events, and the system's evolution depends on the interactions of such events and their arrival times [13, 24]. Data assimilation is also a desired ability in discrete event simulations, especially in real-time applications of simulation models, where the model provides predictions based on the last known state of the system it represents. We take a traffic signal control example to explain the necessities of assimilating data in discrete event simulations. In [5, 17, 19], the authors have presented how traffic signal control systems can be modeled and simulated using discrete event methods; and in reality, we often see the phenomenon that vehicles accumulate at crossings in one direction, while in the orthogonal direction, roads are almost empty with green lights on. Observations of such a situation are very easy to collect by sensors (e.g., inductive loops). If we could assimilate these observations into the discrete event traffic signal control model and dynamically adjust durations of phases of traffic lights, a better performance (e.g., traffic flow) would be achieved.

However, data assimilation in discrete event simulations is not well researched yet, and due to the highly nonlinear, non-Gaussian properties, most data assimilation algorithms cannot be applied in discrete event simulations. Sequential Monte Carlo methods seem to be a set of promising methods which might be applicable in discrete event simulations, since they are able to approximate arbitrary probability densities and have little or no assumption about the properties of the system model [2]. The work in [14, 25] has proven the

SpringSim-TMSDEVS 2016 April 3-6 Pasadena, CA, USA

effectiveness of the SMC methods in discrete event simulations.

A major difficulty of applying the SMC methods is the high computation costs due to the large number of particles, where each particle is represented by a full-scale simulation to the next observation time [2]. To improve the performance of the SMC methods, distributed/parallel SMC methods were developed, such as [2, 23]. In this paper, we propose a novel, efficient SMC method – the *rollback based SMC method*, which is based on the concept of *rollback* [10]. In the proposed method, a particle is a copy (or *partial* copy) of the model state. When there is no data, only one simulation is kept running; while when data is available, the simulation is recursively rolled back to particles generated in the last data assimilation, and then run to the current observation time to generate new particles. After resampling, the particle with the highest probability is assigned to the model, while other particles are kept for the next assimilation. The simulation assigned with the most probable particle is run again until the next observation arrives.

Based on the analysis in section 3.2, the rollback based SMC method consumes much less memory, and is still as fast as its standard counterparts. Due to the space limitations, the performance evaluation is not included in current paper, but will be extended in future. An identical-twin experiment in a traffic case which is implemented in a discrete event traffic simulation software is carried out to evaluate the accuracy of the proposed algorithm. The results show that the simulation with the rollback based SMC method can accurately estimate the position of a slow vehicle and the traffic density on the road.

The remainder of the paper is organized as follows. Section 2 reviews the existing data assimilation methods, and clarifies the research gaps. Section 3 presents the rollback based SMC method, and provides a theoretical analysis of its performance in terms of the memory consumption and speed. Experiments and results are given in section 4. Conclusions are drawn in section 5.

2. RELATED WORK

2.1 Traditional Data Assimilation Algorithms

Traditional data assimilation is conducted based on two models [1]: one is the *system model* which describes the evolution of the state with time, and the other is the *measurement model* which relates the noisy observations to the state. The two models are conventionally expressed as difference equations [16]:

$$\begin{aligned} x_k^t &= M_k(x_{k-1}^t) + \nu_{k-1} \\ y_k^o &= H_k(x_k^t) + \varepsilon_k \end{aligned} \tag{1}$$

where x and M are the model's state vector and its corresponding dynamics operator, respectively; y is the observation vector, and H is a mapping between the state space and the observation space. ν_{k-1} and ε_k respectively model errors in the system model and in the measurement model. Superscript $(\cdot)^t$ denotes the *true* state, while $(\cdot)^o$ denotes the *observation*. It is commonly assumed that ν_{k-1} has a zero mean and a covariance matrix $\mathbf{Q}_{k-1}$, and observations are unbiased and have a covariance matrix $\mathbf{R}_k$ [18].

Information, whether in the form of observations or models, has errors, and data assimilation is a class of algorithms which make objective compromises between these sources of uncertain information based on principles which aim to maximize (or minimize) a quantity (e.g., a penalty function) [18].

There are two distinct classes of data assimilation algorithms: one is the class of *variational techniques*, and the other is the class of *sequential methods*. *3-Dimensional Variational Analysis* (3D-VAR) and *4-Dimensional Variational Analysis* (4D-VAR) are two typical variational techniques in use. 3D-VAR minimizes the cost function J shown in equation (2) which measures the misfit between the model state x and the background state x^b (commonly derived from a short-range forecast), and also between x and the observation y^o. The minimization of J is done with respect to x, and the resultant x is termed the *analysis*, x^a.

$$\begin{aligned} J(x) =& \frac{1}{2}(x - x^b)^T \mathbf{Q}^{-1}(x - x^b) \\ &+ \frac{1}{2}(y^o - H(x))^T \mathbf{R}^{-1}(y^o - H(x)) \end{aligned} \tag{2}$$

In 3D-VAR, all observations in the time-window are treated as if they occurred at the same time [20]. This introduces some errors because real systems are changing and developing. 4D-VAR addresses this problem by introducing the time dimension into assimilation [18]. In 4D-VAR, the observation operators are generalized to include a forecast model that will allow a comparison between the model states and the observations distributed on a time window [3].

Sequential methods assimilate data sequentially in time and their main objective is to correct the estimated variables at every time an observation becomes available [22]. Sequential methods usually consist of a *forecast step*

$$x_k^f = M_k(x_{k-1}^a) \tag{3}$$

, followed by an *update step*

$$x_k^a = x_k^f + \mathbf{K_k}(y_k^o - H_k(x_k^f)) \tag{4}$$

where superscript $(\cdot)^f$ means forecast, and $\mathbf{K}_k = \mathbf{Q_{k-1}}\mathbf{H}_k^T(\mathbf{H}_k\mathbf{Q}_{k-1}\mathbf{H}_k + \mathbf{R}_k)^{-1}$ is a gain matrix which is chosen to minimize the *analysis error covariance matrix* $\mathbf{P}_k^a = \overline{(x_k^t - x_k^a)(x_k^t - x_k^a)^T}$, and $\mathbf{H}_k$ is the tangent linear function of H_k at x_k^f. If M_k and H_k are linear, and ν_{k-1} and ε_k are stationary zero-mean white noise [1], the *forecast error covariance matrix* $\mathbf{P}_k^f = \overline{(x_k^f - x_k^t)(x_k^f - x_k^t)^T}$ and the analysis error covariance matrix $\mathbf{P}_k^a$ can be accurately calculated. This yields to the *optimal* filter – *Kalman Filter* (KF). However, In many situations of interest, the linear and Gaussian assumption does not hold. Non-linear and non-Gaussian conditions have no influence on calculation of the forecast and analysis in equation (3) and (4), but poses great difficulties on computation of error covariance matrices $\mathbf{P}_k^f$ and $\mathbf{P}_k^a$. Many approximation methods are proposed to tackle these difficulties. The *Extended Kalman Filter* (EKF) linearizes the

non-linear models locally around x_k^f to ease the computation of $\mathbf{P}_k^f$ and $\mathbf{P}_k^a$ [12]. Although EKF is effective in many practical cases, the method fails to account for the fully nonlinear dynamics in propagating the error covariance, which, in turn, fails to represent the error probability density [12]. Another approach which tackles nonlinearity and non-Gaussian circumstances very well is the *Ensemble Kalman Filter* (EnKF) [9]. EnKF belongs to a broader category of filters known as *Particle Filters* (a detailed introduction of particle filters is presented in section 2.2). In EnKF, the error covariance matrices are approximated by using an ensemble of model states. Theoretically, full error statistics can be exactly represented by an infinite ensemble of model states [9]. EnKF does not involve an approximation of the nonlinearity of M_k and H_k, the computational burden of evaluating the Jacobians is hence absent [12]. *Optimal Interpolation* (OI) *Analysis* is another suboptimal filter within the general framework of sequential methods. In OI Analysis, $\mathbf{P}_k^f$ is replaced by an approximation, $\mathbf{S}_k^f$ [16], which is computed as a product of a time-independent correlation matrix $\mathbf{C}$ and a diagonal variance matrix $\mathbf{D}_k^f$. $\mathbf{D}_k^f$ is incremented over model time steps by an empirically determined approximation of mean forecast error growth [11]. OI Analysis is particularly efficient when only a few observations are important for each variable in determining the gain in the update step [3].

2.2 Sequential Monte Carlo (SMC) Methods

SMC methods, also called *particle filters*, is a technique for implementing a recursive Bayesian filter by Monte Carlo (MC) simulations [1]. The key idea is to represent the required posterior distribution by a set of random samples (also called *particles*) with associated weights and to compute estimates based on these samples and weights. As the number of samples becomes very large, this MC characterization becomes an equivalent representation to the usual functional description of the posterior distribution [1, 8].

We are interested in obtaining the posterior distribution $p(x_{0:k}|z_{1:k})$, where $x_{0:k} = \{x_i, i = 0, \ldots, k\}$ is the set of all states up to time k, and $z_{1:k} = \{z_i, i = 1, \ldots, k\}$ is the set of all available observations up to the same time. The SMC methods approximate $p(x_{0:k}|z_{1:k})$ by a *random measure*

$$\chi_k = \{x_{0:k}^i, w_k^i\}_{i=1}^N \tag{5}$$

where $\{x_{0:k}^i, i = 1, \ldots, N\}$ is a set of *support points* with associated *weights* $\{w_k^i, i = 1, \ldots, N\}$. The weights are normalized such that $\sum_{i=1}^N w_k^i = 1$. Then we have

$$p(x_{0:k}|z_{1:k}) \approx \sum_{i=1}^{N} w_k^i \delta(x_{0:k} - x_{0:k}^i) \tag{6}$$

where $\delta(\cdot)$ is the Dirac delta function. Usually, direct sampling from $p(x_{0:k}|z_{1:k})$ is intractable, therefore the sequential importance sampling (SIS) algorithm is developed, in which samples are drawn from an easily sampled distribution which is called *importance density*, and weights are updated using

$$w_k^i \propto \frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)}{\pi(x_k^i|x_{0:k-1}^i, z_{1:k})} w_{k-1}^i \tag{7}$$

where $\pi(x_k^i|x_{0:k-1}^i, z_{1:k})$ is the importance density. Readers can refer to [1] and [8] for more details on the derivation of the SIS algorithm. The SIS algorithm can be implemented by performing the following two steps for every k:

1. draw particles $x_k^i \sim \pi(x_k|x_{0:k-1}^i, z_{1:k}), i = 1, \ldots, N$, and append them to $x_{0:k-1}^i$ to form $x_{0:k}^i$;
2. compute the weights w_k^i according to equation (7), and normalize the weights.

The importance density plays a very important role in the performance of the SMC methods. In general, the closer the importance density to that distribution, the better the approximation [8]. If $\pi(x_k|x_{0:k-1}^i, z_{1:k}) = \pi(x_k|x_{k-1}^i, z_k)$, then the importance density becomes only dependent on x_{k-1} and z_k. This is particularly useful in the common case when only a filtered estimate of $p(x_k|z_{1:k})$ is required at each time step. In such scenarios, only x_k^i need be stored, therefore, one can discard the path $x_{0:k-1}^i$ and history of observations $z_{1:k-1}$. The *prior importance density* which is given by $p(x_k|x_{k-1}^i)$ [8] is a such density, and it implies particle weight updates by

$$w_k^i \propto p(z_k|x_k^i) w_{k-1}^i \tag{8}$$

The simplified SIS algorithm is shown in Algorithm 1.

Algorithm 1: The SIS algorithm

Input: random measure at $k-1$:
$\chi_{k-1} = \{x_{k-1}^i, w_{k-1}^i\}_{i=1}^N$, new observation z_k
Output: random measure at k: $\chi_k = \{x_k^i, w_k^i\}_{i=1}^N$
for $i = 1 : N$ **do**
  draw particles $x_k^i \sim p(x_k|x_{k-1}^i, z_k)$;
  assign the particle a weight, w_k^i, according to (8);
end

A major problem with the SIS algorithm is that the discrete random measure degenerates quickly [1, 8]. In other words, all the particles except for a very few are assigned negligible weights. Degeneracy can be reduced by *resampling*, in which particles are replicated in proportion to their weights [8]. A complete implementation of the SMC methods based on the SIS algorithm and resampling is presented in Algorithm 2.

2.3 Data Assimilation in Discrete Event Simulations

In discrete event simulations, the model state and its evolution have very different properties from those in continuous systems and models where traditional data assimilation algorithms are developed and applied:

1. *The behavior of discrete event simulations is highly nonlinear, non-Gaussian*. The functions to describe state evolution in discrete event simulations are usually rule-based. These functions are essentially step functions and can therefore not be linearized, because state changes happen instantaneously at the event. The high nonlinearity of state transition functions in discrete event simulations hampers the application of data assimilation algorithms which are based on linear assumption (KF) or local linearization (EKF).

Algorithm 2: The SMC methods

Input: random measure at $k-1$:
$\chi_{k-1} = \{x^i_{k-1}, w^i_{k-1}\}^N_{i=1}$, new observation z_k
Output: random measure at k: $\chi_k = \{x^i_k, w^i_k\}^N_{i=1}$
% the sampling step
for $i = 1 : N$ **do**
  draw particles $x^{'i}_k \sim p(x_k|x^i_{k-1}, z_k)$;
  assign $x^{'i}_k$ a weight: $w^{'i}_k = p(z_k|x^{'i}_k)w^i_{k-1}$;
end
normalize the weights: $w^{''i}_k = \frac{w^{'i}_k}{\sum_{j=1}^N w^{'j}_k}$
% the resampling step
$c_0 = 0$;
for $i = 1 : N$ **do**
  $c_i = c_{i-1} + w^{''i}_k$
end
for $i = 1 : N$ **do**
  generate a random number $r \sim U[0, 1]$;
  if $c_{j-1} < r \leq c_j$ **then**
    $x^i_k = x^{'j}_k$;
    $w^i_k = \frac{1}{N}$;
  end
end

2. *The model state in discrete event simulations can hardly be represented as a meaningful vector*. In continuous systems and models, the system state is represented as a column vector x [3], and elements of the state vector usually have same meaning and similar referents, for example, a two-dimension state vector $x = [x_T\ x_M]^T$ represents temperatures in Toronto and Montreal, respectively. The state trajectory is continuously changing with the state taking value in $\mathbb{R}^n$ [13]. In discrete event simulations, however, the state trajectory is piecewise constant and event driven [13], and state elements of an atomic model are rather diverse, for example, (s, e) is the total state of a barber model, where s is a *boolean* type which indicates whether the barber is busy or not, and e is a *double* type which tells how long the barber has been in the current state. For a coupled model with hierarchical structure, the model state is usually defined as a collection of states of all its atomic components. Therefore, in discrete event simulations, it is almost impossible to represent the model state as a meaningful vector, thus error covariance matrices, such as $\mathbf{Q}$ and $\mathbf{R}$, are not available anymore. These matrices are essential to almost all algorithms introduced in section 2.1, because they are used to compute the gain matrix $\mathbf{K}$, forecast error covariance matrix $\mathbf{P}^f$, and analysis error covariance matrix $\mathbf{P}^a$, etc.

Therefore, the algorithms introduced in section 2.1 can hardly be applied in discrete event simulations.

Sequential Monte Carlo methods seem to be a set of promising methods which are applicable in discrete event simulations, since they are able to approximate arbitrary probability densities and have little or no assumption about the properties of the system model [2]. The work in [14, 25] has proven the effectiveness of the SMC methods in discrete event simulations.

A major difficulty of applying the SMC methods is the high computation costs due to the large number of particles, where each particle is represented by a full-scale simulation to the next observation time [2]. To improve the performance of the SMC methods, distributed/parallel SMC methods have been developed, such as [2, 7, 23]. In this paper, we propose a novel, efficient SMC method – the *rollback based SMC method*, which is based on the concept of *rollback* [10]. In the proposed algorithm, a particle is a copy (or *partial* copy) of the model state. When there is no data, only one simulation is kept running; while when data is available, the simulation is first rolled back to a particle in the last data assimilation, and then run to the current observation time. At this moment, a particle is generated by retrieving the model state. This procedure is repeated until the specified number of particles are generated. After resampling, the particle with the highest probability is assigned to the model, while other particles are kept for the next assimilation. The simulation assigned with the most probable particle is run again until the next observation arrives. Therefore, the proposed algorithm consumes much less memory, and we will prove that it is still as fast as its standard counterparts. Due to the space limitations, this paper mainly focuses on the description of the rollback based SMC method, and its effectiveness proof through a traffic case, which are presented in section 3.1 and section 4, respectively, while the memory consumption and speed comparison are only analyzed briefly in section 3.2, and related experiment will be extended in future.

3. ROLLBACK BASED SMC METHOD

3.1 Algorithm Description

The main idea of the rollback based SMC method is shown in Figure 1. In our proposed method, N particles are also employed, but different with the standard SMC methods where each particle is evolved by an independent, full-scale simulation to the next observation time, a particle in our method is a copy (or *partial* copy) of the model state, and only one simulation is kept for generating particles.

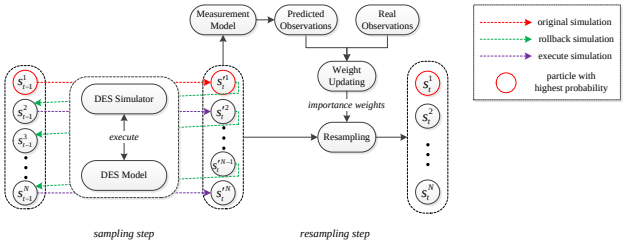

Figure 1. The rollback based SMC method

In a set of particles $\{s^i_t, i = 1, \ldots, N\}$, we always assume that the first particle s^1_t has the highest probability. Assume the initial state is distributed as $p(s_0)$. In the initialization, N particles $\{s^i_0, i = 1, \ldots, N\}$ are drawn from $p(s_0)$, but

only s_0^1 is assigned to the model and the simulation starts to run, i.e., only one simulation with the most probable state is running.

Assume at time t, observation o_t is collected, and the particles at the last data assimilation time $t-1$ are $\{s_{t-1}^i, i = 1, \ldots, N\}$. The sampling step generates N particles $\{s_t^{'i}, i = 1, \ldots, N\}$ as follows:

1. the first particle $s_t^{'1}$ is already embedded in the simulation, therefore, it can be generated by retrieving the current state of the model;
2. the other $N-1$ particles $\{s_t^{'i}, i = 2, \ldots, N\}$ are generated recursively as follows:
   (a) roll back the current simulation to s_{t-1}^i;
   (b) run the simulation to time t, and $s_t^{'i}$ is generated by retrieving the model state.

The resampling step in the proposed algorithm is the same with that in the standard SMC methods, therefore, it will not be repeated here. After resampling, the particle with the highest probability is assigned to the model, while other particles are kept for the next assimilation. The simulation assigned with the most probable particle is run again until the next observation arrives.

In section 4, a discrete event traffic case is studied to prove the effectiveness of the proposed algorithm. The traffic model is built in OpenTrafficSim[1], which is a Java based, open source, discrete event simulation software to support research and development of multi scale and multi modal traffic models. In order to do rollback in OpenTrafficSim, the vehicle class is extended to enable the discrete event model to save and restore its state; the discrete event simulator is revised to be capable of setting its time backward. Besides, after the model state and simulator time are rolled back, the model reschedules events on its simulator to reconstruct the future event list.

3.2 Memory Consumption & Speed

In this section, we briefly analyze the memory consumption and speed of the rollback based SMC method compared with the standard SMC methods. The analysis shows that the rollback based SMC method consumes much less memory, but is almost as fast as the standard SMC methods.

Memory Consumption

Assume the memory consumption of a discrete event model is M_m, while the proportion of the memory consumption of the state in the whole memory consumption of the model is $p(0 < p \leq 1)$. The memory consumption of a discrete event simulator with an empty event list is M_s. During the simulation, we assume there are N_e events on average stored in the event list, and the average memory consumption of an event is M_e. In the standard SMC methods with N particles, the memory consumption at time t is

$$MC_{standard} = N(M_m + M_s + N_e M_e)$$

[1]More information about OpenTrafficSim can be found in **http://www.opentrafficsim.org/**.

, while in the rollback based SMC method with the same number of particles, the memory consumption is

$$MC_{rollback} = M_m + M_s + N_e M_e + (N-1)M_m p$$

Obviously, $MC_{rollback} \ll MC_{standard}$, i.e., the rollback based SMC method consumes much less memory compared with the standard SMC methods.

Speed

In order to accelerate the execution of the SMC methods, parallel/distributed SMC methods are commonly applied. If there is only one processing unit (PU) available, N particles in the standard SMC methods can only be executed sequentially, therefore, in the single-PU case, the rollback based SMC method is almost as fast as the standard SMC methods.

If there are multiple PUs available (assume the number of PUs is N_p, usually we have $N_p < N$), the rollback based SMC method can be parallelized or distributed as follows: deploy one simulation and n_i particles on the processing unit $PU_i (i = 1, \ldots, N_p)$, such that $\sum_{i=1}^{N_p} n_i = N$; on PU_i, particles will evolve as described in section 3.1; when all particles are generated, resampling will be done in a centralized way. Therefore, in the multiple-PU case, there is no obvious speed drop in the rollback SMC method compared with the standard one if proper parallelizing/distributing schemes are employed.

4. EXPERIMENTS AND RESULTS

4.1 Experiment Setup

The identical-twin experiment is adopted in this paper to evaluate the effectiveness of the proposed rollback based SMC method. In the experiment, we first model a two-lane circular road shown in Figure 2 in OpenTrafficSim. The total length of the road is 2000 meters, on which 80 vehicles are driving in counterclockwise direction. Along the road, 8 sensors are evenly installed, and each sensor has 100-meter detection range on each lane. Each sensor can report two types of information every 10 seconds: 1) the number of vehicles in its detection range; 2) the average speed of all vehicles in the detection range (if there is no vehicle in the range, the speed is defined as the maximum allowed speed of the road). The experiment consists of three traffic simulations which have the same run length of 1000 seconds:

1. *'real' traffic*: the traffic simulation from which the real data is obtained. Among 80 vehicles on the road, we assume there is one slower vehicle ($v_{max} = 25km/h$, which is $1/4$ of the normal vehicles' maximum speed). Besides the sensor readings which are collected every 10 seconds, two other types of information are recorded for later comparison: 1) slow vehicle position; 2) number of vehicles on each segment (each lane is evenly divided into 20 segments).
2. *simulated traffic*: the traffic simulation which has no information about the real traffic, therefore, we randomly choose one vehicle as the slow vehicle.

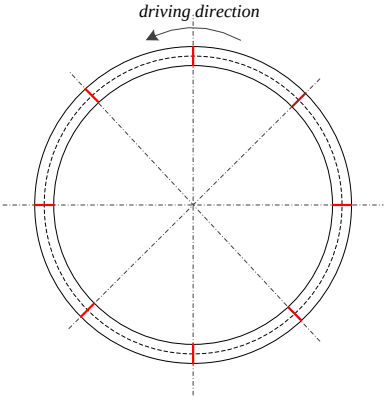

Figure 2. The two-lane circular road, sensors are evenly installed at positions marked in red

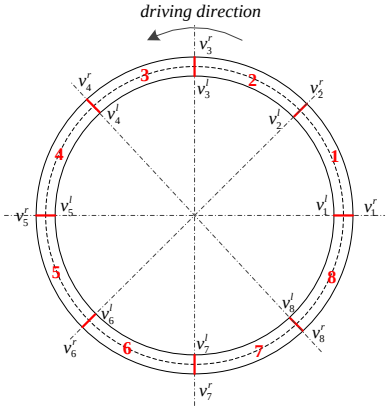

Figure 3. Estimating the slow vehicle position by average speed increases between two consecutive sensors

3. *filtered traffic*: the traffic simulation which has the same configuration with the simulated traffic, except that the sensor readings are assimilated by the rollback based SMC method.

In the filtered traffic, we use $N = 60$ particles. A particle is a collection of vehicle information, including *position*, *velocity*, *acceleration*, and *maximum speed* (used for recognizing the slow vehicle). The simulation is initialized with the same configuration with the simulated traffic, and the other $N-1$ particles are generated by randomly selecting a vehicle in each particle and setting its maximum speed as $25km/h$. When generating particles, we add random noise to the simulation just before we retrieve its state when the simulation runs to the observation time. The random noise is to select a normal vehicle with a probability of 0.1 around the slow vehicle (50 meters before and after) and exchange their maximum speed.

Weight computation is a very important step in SMC methods since it provides the measure to keep the optimal particles for future steps. In this paper, the weight of the i-th particle is computed by taking three types of information into consideration:

1. number of vehicles in each sensor's detection range. We define $d_i = max(\max_{j=1,\dots,8} |n_j^l - n_j^{'l}|, \max_{j=1,\dots,8} |n_j^r - n_j^{'r}|)$, where n_j^l, n_j^r are the numbers of vehicles on the left and right lane respectively within the detection range of j-th sensor in the real traffic, while $n_j^{'l}$ and $n_j^{'r}$ are the corresponding numbers in the particle. The contribution of this information to the weight is defined as
$$p_1^i = \frac{1}{\sqrt{2\pi}} e^{-\frac{d_i^2}{2}}$$

2. average speed of vehicles in each sensor's detection range. Similar to the way in dealing with the number of vehicles, we assume v_i as the maximum average speed difference between the real traffic and the particle, and define its contribution as
$$p_2^i = \frac{1}{\sqrt{2\pi}} e^{-\frac{v_i^2}{2}}$$

3. the difference of slow vehicle positions between the real traffic and the particle. Because we have no information about the slow vehicle position in the real traffic, we need to estimate the area the slow vehicle is probably in. A slow vehicle can block vehicles behind it, hence, there will be a sharp increase of average speed around the slow vehicle. We calculate the speed increase between two consecutive sensors by
$$I_j^l = \begin{cases} v_{j+1}^l - v_j^l & j = 1, \dots, 7 \\ v_1^l - v_8^l & j = 8 \end{cases}$$
where v_j^l is the average speed of vehicles on the left lane within the detection range of the j-th sensor. Assume j^{*l} is the index of sensor which satisfies $I_{j^{*l}}^l = \max_{j=1,\dots,8} I_j^l$, therefore, the slow vehicle should be in the area j^{*l} shown in Figure 3. But in order to be more robust, we extend the area by adding two adjacent areas: area j_b^{*l} and area j_f^{*l}, which is behind and in the front of area j^{*l}, respectively. Suppose in the particle, the slow vehicle is in area k, we define
$$p_l = \begin{cases} 0.9 & k = j^{*l} \\ 0.05 & k = j_b^{*l} \text{ or } j_f^{*l} \end{cases}$$
The same method applies to the data of the right lane, then we can get j^{*r}, j_b^{*r}, j_f^{*r} and p_r. The contribution to the weight is therefore defined by
$$p_3^i = (p_l + p_r)/2$$

The three contributions are linearly combined as $p^i = \alpha p_1^i + \beta p_2^i + \gamma p_3^i$, such that $\alpha + \beta + \gamma = 1$. In our experiment, we choose $\alpha = \beta = 0.25, \gamma = 0.5$. Then the weight of the i-th particle is accordingly updated by $w_t^{'i} = w_{t-1}^i p^i$, and is finally normalized by $w_t^i = w_t^{'i} / \sum_{j=1}^N w_t^{'j}$.

4.2 Experiment Results

The filtered traffic provides estimations of the *slow vehicle position* and the *traffic density* on an arbitrarily chosen road segment by assimilating sensor readings from the real traffic using the rollback based SMC method. The results show that

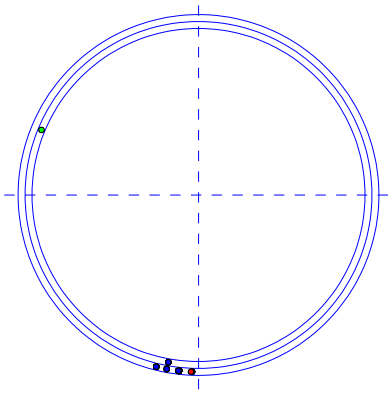

Figure 4. Estimation of the slow vehicle position ($t = 1000$); the slow vehicle positions in the real traffic, the simulated traffic and the filtered traffic are marked in red, green and blue, respectively

the filtered traffic provides accurate estimations of the slow vehicle position and the traffic density on the road. The experiment proves that the rollback based SMC method is an effective data assimilation algorithm.

Locate the slow vehicle

The slow vehicle in real traffic at time 1000 is shown in red in Figure 4, while the estimations given by the simulated traffic and the filtered traffic at the same time are shown in green and blue, respectively. The distance between the true position and the estimation given by the simulated traffic is $622.93m$, while it decreases to $30.24m$ when real data is fed into the simulation by the rollback based SMC method.

Estimate the traffic density

We arbitrarily choose one segment (length of $100m$) of a lane (the inside lane of the circular road), and estimate the traffic density over time. The result presented in Figure 5(a) shows that the simulated traffic can only estimate accurately in the beginning ($t \leq 170$), but fails to generate accurate estimation as the time evolves; while as shown in Figure 5(b), although the estimation is not accurate all the time, the filtered traffic can accurately capture the trend of the density evolving, i.e., when the segment is busy and when it becomes idle, and can also roughly give an acceptable estimation of how busy it is.

We define $E_1 = \frac{1}{N_T}\sum_{i=1}^{N_T}|N_i^r - N_i|$ as the estimation error of the simulated traffic, where N_T is the number of data points (in our experiment, $N_T = 1000/10 = 100$); N_i^r and N_i are numbers of vehicles on the chosen segment in the real traffic and the simulated traffic at time t_i, respectively. Similarly, we can define the estimation error of the filtered traffic as $E_1' = \frac{1}{N_T}\sum_{i=1}^{N_T}|N_i^r - N_i'|$, where N_i' is the number of vehicles on the chosen segment in the filtered traffic. The experiment reveals that the estimation error of the simulated traffic is $E_1 = 2.3168$, but it shows a 55.2046% drop ($E_1' = 1.0378$) when the real data is assimilated by the rollback based SMC method.

Figure 6 shows the traffic densities of different road segments (the circular road is evenly divided into 20 segments) at time 1000. The result shows that the simulated traffic fails to estimate the traffic densities on different road segments, while

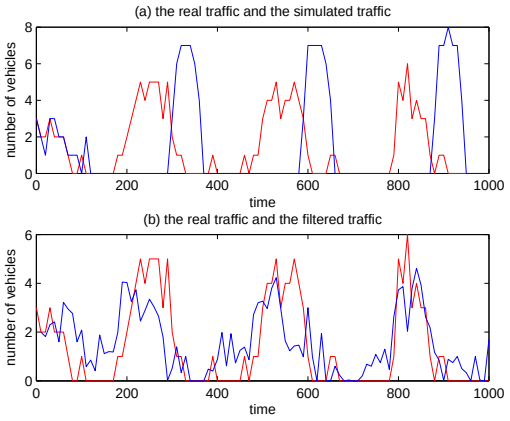

Figure 5. Number of vehicles on one segment ($100m$) of one lane. The red line shows the number of vehicles in the real traffic, while the blue line in (a) shows the number of vehicles in the simulated traffic, and the blue line in (b) shows the number of vehicles in the filtered traffic

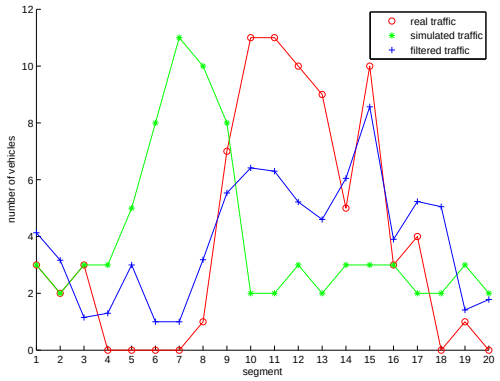

Figure 6. Number of vehicles on different segments on the road ($t = 1000$)

the filtered traffic can roughly reconstruct the traffic situation on the road. If we define the estimation error of the simulated traffic as $E_2 = \frac{1}{N_s}\sum_{i=1}^{N_s}|N_i^r - N_i|$, where N_s is the number of road segments, N_i^r and N_i are numbers of vehicles on the i-th road segment in the real traffic and the simulated traffic, respectively. Similarly, we can define the estimation error of the filtered traffic as $E_2' = \frac{1}{N_s}\sum_{i=1}^{N_s}|N_i^r - N_i'|$, where N_i' is the number of vehicles on the i-th road segment in the filtered traffic. The experiment shows that the estimation error of the filtered traffic is $E_2' = 2.2217$, while it goes up to $E_2 = 4.3$ (increased by 48.3333%) if no data is assimilated into the simulation.

5. CONCLUSION

This paper presents a novel and efficient data assimilation algorithm – the rollback based SMC method. Different from its standard counterparts where each particle is evolved by an independent, full-scale simulation to the next observation time, a particle in our method is a copy (or *partial* copy) of the model state, and only one simulation is kept for generating the particles by exploiting the rollback technique. A theoretical analysis shows that the proposed algorithm consumes

much less memory, but is almost as fast as the standard SMC methods. Due to space limitations, the experiment on the performance evaluation is not included in this paper, but will be extended in future. An identical-twin experiment in a discrete event traffic case is carried out to evaluate the accuracy of the proposed algorithm. The results show that the simulation with the rollback based SMC method can accurately estimate the slow vehicle position and the traffic density on the road.

Acknowledgment

This research is mainly financed by the China Scholarship Council (Grant NO. 201306110027), and two National Natural Science Foundations of China (Grant NO. 61374185 and 61403402, respectively).

REFERENCES

1. Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing 50*, 2 (2002), 174–188.
2. Bai, F., Gu, F., Hu, X., , and Guo, S. Particle routing in distributed particle filters for large-scale spatial temporal systems. *IEEE Transactions on Parallel and Distributed Systems* (2015), to appear.
3. Bouttier, F., and Courtier, P. Data assimilation concepts and methods. *Meteorological Training Course Lecture Series, ECMWF* (1999).
4. Carton, J. A., and Giese, B. S. A reanalysis of ocean climate using simple ocean data assimilation (SODA). *Monthly Weather Review 136* (2008), 2999–3017.
5. Chi, S.-D., Lee, J.-O., and Kim, Y.-K. Discrete event modeling and simulation for traffic flow analysis. In *IEEE International Conference on Systems, Man and Cybernetics*, vol. 1 (1995), 783–788.
6. Constantinescu, E. M., Sandu, A., Chai, T., and Carmichael, G. R. Ensemble-based chemical data assimilation. I: General approach. *Quarterly Journal of the Royal Meteorological Society 133*, 626 (2007), 1229–1243.
7. Darema, F. Dynamic data driven applications systems: A new paradigm for application simulations and measurements. In *Computational Science - ICCS 2004*, M. Bubak, G. Albada, P. M. Sloot, and J. Dongarra, Eds., vol. 3038 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004, 662–669.
8. Djurić, P. M., Kotecha, J. H., Zhang, J., Huang, Y., Ghirmai, T., Bugallo, M. F., and Miguez, J. Particle filtering. *IEEE Signal Processing Magazine 20*, 5 (2003), 19–38.
9. Evensen, G. The ensemble Kalman filter: theoretical formulation and practical implementation. *Ocean Dynamics 53*, 4 (2003), 343–367.
10. Fujimoto, R. M. *Parallel and Distributed Simulation Systems*. Wiley New York, 2000.
11. Ghil, M., and Malanotte-Rizzoli, P. Data assimilation in meteorology and oceanography. *Advances in Geophysics 33* (1991), 141–266.
12. Gillijns, S., Mendoza, O., Chandrasekar, J., De Moor, B. L. R., Bernstein, D., and Ridley, A. What is the ensemble Kalman filter and how well does it work? In *American Control Conference* (2006), 4448–4453.
13. Ho, Y.-C. Introduction to special issue on dynamics of discrete event systems. *Proceedings of the IEEE 77*, 1 (1989), 3–6.
14. Hu, X. Dynamic data driven simulation. *SCS M&S Magazine II*, 1 (2011), 16–22.
15. Huang, X.-Y., Xiao, Q., Barker, D. M., Zhang, X., Michalakes, J., Huang, W., Henderson, T., Bray, J., Chen, Y., Ma, Z., Dudhia, J., Guo, Y., Zhang, X., Won, D.-J., Lin, H.-C., and Kuo, Y.-H. Four-dimensional variational data assimilation for WRF: Formulation and preliminary results. *Monthly Weather Review 137*, 1 (2009), 299–314.
16. Ide, K., Courtier, P., Ghil, M., and Lorenc, A. C. Unified notation for data assimilation : Operational, sequential and variational. *Journal of the Meteorological Society of Japan, Special Issue on "Data Assimilation in Meteology and Oceanography: Theory and Practice" 75*, 1B (1997), 181–189.
17. Kang, D., Kong, J., and Choi, B. K. DEVS modeling of urban traffic systems (WIP). In *Proceedings of the 2012 Symposium on Theory of Modeling and Simulation - DEVS Integrative M&S Symposium* (2012), 16:1–16:6.
18. Lahoz, W. A., and Schneider, P. Data assimilation: Making sense of earth observation. *Frontiers in Environmental Science 2*, 16 (2014), 1–28.
19. Lee, J., and Chi, S. Using symbolic DEVS simulation to generate optimal traffic signal timings. *Simulation: Transactions of the Society for Modeling and Simulation International 81*, 2 (2005), 153–170.
20. Lorenc, A. C., and Rawlins, F. Why does 4D-Var beat 3D-Var? *Quarterly Journal of the Royal Meteorological Society 131*, 613 (2005), 3247–3257.
21. Nichols, N. K. Data assimilation: aims and basic concepts. In *Data Assimilation for the Earth System*, R. Swinbank, V. Shutyaev, and W. Lahoz, Eds., vol. 26 of *NATO Science Series*. Springer Netherlands, 2003, 9–20.
22. Pelc, J. S. *Data Assimilation for Marine Ecosystem Models*. PhD thesis, Delft University of Technology, Department of Applied Mathematics, Delft, the Netherlands, July 2013.
23. Teulière, V., and Brun, O. Parallelisation of the particle filtering technique and application to doppler-bearing tracking of maneuvering sources. *Parallel Computing 29*, 8 (2003), 1069–1090.
24. Wainer, G. A. *Discrete-Event Modeling and Simulation: A Practitioner's Approach*. CRC Press, 2009.
25. Xue, H., Gu, F., and Hu, X. Data assimilation using sequential Monte Carlo methods in wildfire spread simulation. *ACM Transactions on Modeling and Computer Simulation 22*, 4 (2012), 23:1–23:25.