Event-Scheduling Algorithms with Kalikow Decomposition for Simulating Potentially Infinite Neuronal Networks

Tien Cuong Phi1 · Alexandre Muzy2 · Patricia Reynaud-Bouret1

Received: 22 July 2019 / Accepted: 2 September 2019 / Published online: 19 October 2019 © Springer Nature Singapore Pte Ltd 2019

Abstract
Event-scheduling algorithms can compute in continuous time the next occurrence of points (as events) of a counting process based on their current conditional intensity. In particular, event-scheduling algorithms can be adapted to perform the simulation of finite neuronal networks activity. These algorithms are based on Ogata’s thinning strategy (Ogata in IEEE Trans Inf Theory 27:23–31, 1981), which always needs to simulate the whole network to access the behavior of one particular neuron of the network. On the other hand, for discrete time models, theoretical algorithms based on Kalikow decomposition can pick at random influencing neurons and perform a perfect simulation (meaning without approximations) of the behavior of one given neuron embedded in an infinite network, at every time step. These algorithms are currently not computationally tractable in continuous time. To solve this problem, an event-scheduling algorithm with Kalikow decomposition is proposed here for the sequential simulation of point processes neuronal models satisfying this decomposition. This new algorithm is applied to infinite neuronal networks whose finite time simulation is a prerequisite to realistic brain modeling.

Keywords Kalikow decomposition · Discrete event simulation · Point process · Infinite neuronal networks

Introduction

Point processes in time are stochastic objects that model efficiently event occurrences with a huge variety of applications: time of deaths, earthquake occurrences, gene positions on DNA strand, etc. [1, 20, 22]).

Most of the time, point processes are multivariate [7] in the sense that either several processes are considered at the same time, or in the sense that one process regroups together all the events of the different processes and marks them by their type. A typical example consists in considering either two processes, one counting the wedding events of a given person and one counting the children birth dates of the same person or only one marked process which regroups all the possible dates of birth or weddings independently and adds one mark per point, here wedding or birth.

Consider now a network of neurons each of them emitting action potentials (spikes). These spike trains can be modeled by a multivariate point process with a potentially infinite number of marks, each mark representing one given neuron. The main difference between classical models of multivariate point processes and the ones considered in particular for neuronal networks is the size of the network. A human brain consists in about $10^{11}$ neurons whereas a cockroach contains already about $10^6$ neurons. Therefore, the simulation of the whole network is either impossible or a very difficult and computationally intensive task for which particular tricks depending on the shape of the network or the point processes have to be used [6, 14, 19].

Another point of view, which is the one considered here, is to simulate, not the whole network, but the events of one particular node or neuron, embedded in and interacting with the whole network. In this sense, one might consider an infinite network. This is the mathematical point of view considered in a series of papers [9, 10, 18] and based on

This article is part of the topical collection “Modelling methods in Computer Systems, Networks and Bioinformatics” guest edited by Erol Gelenbe.

Patricia Reynaud-Bouret reynaudb@unice.fr
Tien Cuong Phi cuongphi@unice.fr
Alexandre Muzy muzy@i3s.unice.fr

1 Université Côte d’Azur, CNRS, LJAD, Nice, France
2 Université Côte d’Azur, CNRS, I3S, Nice, France
Kalikow decomposition [12] coupled with perfect simulation theoretical algorithms [4, 8]. However, these works are suitable in discrete time and only provide a way to decide at each time step if the neuron is spiking or not. They cannot operate in continuous time, i.e., they cannot directly predict the next event (or spike). Up to our knowledge, there exists only one attempt of using such decomposition in continuous time [11], but the corresponding simulation algorithm is purely theoretical in the sense that the corresponding conditional Kalikow decomposition should exist given the whole infinite realization of a multivariate Poisson process, with an infinite number of marks, quantity which is impossible to simulate in practice.

The aim of the present work is to present an algorithm which

– can operate in continuous time in the sense that it can predict the occurrence of the next event. In this sense, it is an event-scheduling algorithm;
– can simulate the behavior of one particular neuron embedded in a potentially infinite network without having to simulate the whole network;
– is based on an unconditional Kalikow decomposition and, in this sense, can only work for point processes with this decomposition.

In “Event-Scheduling Simulation of Point Processes”, we specify the links between event-scheduling algorithms and the classical point process theory. In “Kalikow Decomposition”, we give the Kalikow decomposition. In “Backward Forward Algorithm”, we present the backward–forward perfect simulation algorithm and prove why it almost surely ends under certain conditions. In “Illustration”, we provide simulation results and a conclusion is given in “Conclusion”.

Event-Scheduling Simulation of Point Processes

On the one hand, simulation algorithms of multivariate point processes [17] are quite well known in the statistical community but as far as we know quite confidential in the simulation (computer scientist) community. On the other hand, event-scheduling simulation first appeared in the mid-1960s [21] and was formalized as discrete event systems in the mid-1970s [23] to interpret very general simulation algorithms scheduling “next events”. A basic event-scheduling algorithm “jumps” from one event occurring at a time stamp \( t \in \mathbb{R}_0^+ \) to a next event occurring at a next time stamp \( t' \in \mathbb{R}_0^+ \) with \( t' \geq t \). In a discrete event system, the state of the system is considered as changing at times \( t, t' \) and conversely unchanging in between [24]. In [14], we have written the main equivalence between the point processes simulation algorithms and the discrete event simulation setup, which led us to a significant improvement in terms of computational time when huge but finite networks are into play. Usual event-scheduling simulation algorithms have been developed considering independently the components (nodes) of a system. Our approach considers new algorithms for activity tracking simulation [5]. The event activity is tracked from active nodes to children (influencers).

Here, we just recall the main ingredients that are useful for the sequel.

To define point processes, we need a filtration or history \((\mathcal{F}_t)_{t \geq 0}\). Most of the time, and this will be the case here, this filtration (or history) is restricted to the internal history of the multivariate process \((\mathcal{F}_t^{\text{int}})_{t \geq 0}\), which means that at time \( t \), i.e., just before time \( t \), we only have access to the events that have already occurred in the past strictly before time \( t \), in the whole network. The conditional intensity, \( \phi_i(t | \mathcal{F}_t^{\text{int}}) \), of the point process representing neuron \( i \), gives the instantaneous firing rate, that is the frequency of spikes, given the past contained in \( \mathcal{F}_t^{\text{int}} \). Let us just mention two very famous examples.

If \( \phi_i(t | \mathcal{F}_t^{\text{int}}) \) is a deterministic constant, say \( M \), then the spikes of neuron \( i \) form a homogeneous Poisson process with intensity \( M \). The occurrence of spikes is completely independent from what occurs elsewhere in the network and from the previous occurrences of spikes of neuron \( i \).

If we denote by \( \mathbf{I} \) the set of neurons, we can also envision the following form for the conditional intensity:

\[
\phi_i(t | \mathcal{F}_t^{\text{int}}) = v_i + \sum_{j \in \mathbf{I}} w_{ij} (\text{Nb}_j^{i}|_{[t-A,t]} \wedge M). \tag{1}
\]

This is a particular case of generalized Hawkes processes [3]. More precisely \( v_i \) is the spontaneous rate (assumed to be less than the deterministic upper bound \( M > 1 \)) of neuron \( i \). Then, every neuron in the network can excite neuron \( i \): more precisely, one counts the number of spikes that have been produced by neuron \( j \) just before \( t \), in a window of length \( A \), this is \( \text{Nb}_j^{i]|_{[t-A,t]} \) we clip it by the upper bound \( M \) and modulate its contribution to the intensity by the positive synaptic weight between neuron \( i \) and neuron \( j \), \( w_{ij} \). For instance, if there is only one spike in the whole network just before time \( t \), and if this happens on neuron \( j \), then the intensity for neuron \( i \) becomes \( v_i + w_{ij} \). The sum over all neurons \( j \) mimics the synaptic integration that takes place at neuron \( i \). As a renormalization constraint, we assume that \( \sup_{t \in \mathbf{I}} \sum_{j \in \mathbf{I}} w_{ij} < 1 \). This ensures in particular that such a process has always a conditional intensity bounded by \( 2M \).

Hence, starting for instance at time \( t \), and given the whole past, one can compute the next event in the network by computing for each node of the network the next event in absence of other spike apparition. To do so, remark that in the absence of other spike apparition, the quantity \( \phi_i(s | \mathcal{F}_s^{\text{int}}) \) for \( s > t \) becomes for instance in the previous example
$\phi^\text{abs}_i(s, t) = v_i + \sum_{j \in I} w_{ij}(\text{Nb}_{i-A,j} \land M)$,

meaning that we do not count the spikes that may occur after $t$ but before $s$. This can be generalized to more general point processes. The main simulation loop is presented in Algorithm 1

Algorithm 1 Classical point process simulation algorithm

▷ With $[t_0, t_1]$ the interval of simulation
1: Initialize the family of points $P = \emptyset$

▷ Each point is a time $T$ with a mark, $j_T$, which is the neuron on which $T$ appears
2: Initialize $t \leftarrow t_0$
3: repeat
4:   for each neuron $i \in I$ do
5:      Draw independently an exponential variable $E_i$ with parameter 1
6:      Apply the inverse transformation, that is, find $T_i$ such that
7:      \[ \int_t^{T_i} \phi^\text{abs}_i(s, t)ds = E_i. \]
8:      Compute the time $T$ of the next spike of the system after $t$, and the neuron where the spike occurs by $T \leftarrow \min_{i \in I} T_i$, with $j_T \leftarrow \arg \min_{i \in I} T_i$
9:      if $T \leq t_1$ then
10:         append $T$ with mark $j_T$ to $P$
11:      end if
12:      $t \leftarrow T$
13: until $t > t_1$

Note that the quantity $\phi^\text{abs}_i(s, t)$ can be also seen as the hazard rate of the next potential point $T_i^{(1)}$ after $t$. It is a discrete event approach with the state corresponding to the function $\phi^\text{abs}_i(\cdot, t)$.

Ogata [17], inspired by Lewis’ algorithm [13], added a thinning (also called rejection) step on top of this procedure because the integral $\int_t^{T_i} \phi^\text{abs}_i(s, t)ds$ can be very difficult to compute. To do so (and simplifying a bit), assume that $\phi_i(t | F_{\cdot,t}^{\text{ext}})$ is upper bounded by a deterministic constant $M$. This means that the point process has always less points than a homogeneous Poisson process with intensity $M$. Therefore, Steps 5–6 of Algorithm 1 can be replaced by the generation of an exponential of parameter $M, E'_i$ and deciding whether we accept or reject the point with probability $\phi^\text{abs}_i(t + E'_i, t)/M$. There are a lot of variants of this procedure: Ogata’s original one uses actually the fact that the minimum of exponential variables is still an exponential variable. Therefore, one can propose a next point for the whole system, then accept it for the whole system and then decide on which neuron of the network the event is actually appearing. More details on the multivariate Ogata’s algorithm can be found in [14].

As we see here, Ogata’s algorithm is very general but clearly needs to simulate the whole system to simulate only one neuron. Moreover, starting at time $t_0$, it does not go backward and, therefore, cannot simulate a Hawkes process in stationary regime. There have been specific algorithms based on cluster representation that aim at perfectly simulating particular univariate Hawkes processes [16]. The algorithm that we propose here will also overcome this flaw.

Kalikow Decomposition

Kalikow decomposition relies on the concept of neighborhood, denoted by $\nu$, which is picked at random and which gives the portion of time and neuron subsets that we need to look at, to move forward. Typically, for a positive constant $A$, such a $\nu$ can be

- $\{(i, [−A, 0])\}$, meaning we are interested only by the spikes of neuron $i$ in the window $[−A, 0)$;
- $\{(i, [−2A, 0]), (j, [−2A, −A])\}$, that is, we need the spikes of neuron $i$ in the window $[−2A, 0)$ and the spikes of neuron $j$ in the window $[−2A, −A)$;
- the emptyset $\emptyset$, meaning that we do not need to look at anything to pursue.

We need to also define $l(\nu)$ the total time length of the neighborhood $\nu$ whatever the neuron is. For instance, in the first case, we find $l(\nu) = A$, in the second $l(\nu) = 3A$ and in the third $l(\nu) = 0$.

We are only interested by stationary processes, for which the conditional intensity, $\phi_i(t | F_{\cdot,t}^{\text{ext}})$, only depends on the intrinsic distance between the previous points and the time $t$ and not on the precise value of $t$ per se. In this sense, the
rule to compute the intensity may be only defined at time 0 and then shifted by \( t \) to have the conditional intensity at time \( t \). In the same way, the timeline of a neighborhood \( \nu \) is defined as a subset of \( \mathbb{R}^* \) so that information contained in the neighborhood is included in \( \mathcal{F}_{0-}^\text{int} \), and \( \nu \) can be shifted (meaning its timeline is shifted) at position \( t \) if need be. We note also that \( \phi^\nu_i \), because it depends on what happens in an empty neighborhood, is a pure constant.

The interpretation of (2) is tricky and is not as straightforward as in the discrete case (see [18]). The best way to understand it is to give the theoretical algorithm for simulating the next event on neuron \( i \) after time \( t \) (cf. Algorithm 2).

---

**Algorithm 2 Kalikow theoretical simulation algorithm**

\(\triangleright\) With \([t_0, t_1]\) the interval of simulation for neuron \( i \)

1. Initialize the family of points \( P = \emptyset \)

\(\triangleright\) NB: since we are only interested by points on neuron \( i \), \( j_T = i \) is a useless mark here.

2. Initialize \( t \leftarrow t_0 \)
3. repeat
4. Draw an exponential variable \( E \) with parameter \( M \), and compute \( T = t + E \).
5. Pick a random neighborhood according to the distribution \( \lambda_i(\cdot) \) given in the Kalikow decomposition and shift the neighborhood at time \( T \); this is \( V_T \).
6. Draw \( X_T \) a Bernoulli variable with parameter \( \phi^\nu_i (T) \)
7. if \( X_T = 1 \) and \( T \leq t_1 \) then
8. append \( T \) to \( P \)
9. end if
10. \( t \leftarrow T \)
11. until \( t > t_1 \)

This Algorithm is close to Algorithm 1 but adds a neighborhood choice (Step 5) with a thinning step (Steps 6–9). In Appendix A, we prove that this algorithm indeed provides a point process with an intensity given by (2) shifted at time \( t \).

The previous algorithm cannot be put into practice because the computation of \( \phi^\nu_i \) depends on the points in \( V_T \) that are not known at this stage. That is why the efficient algorithm that we propose in the next section goes backward in time before moving forward.

### Backward Forward Algorithm

Let us now describe the complete Backward Forward algorithm (cf. Algorithm 3). Note that to do so, the set of points \( P \) is not reduced, as in the two previous algorithms, to the set of points that we want to simulate but this contains all the points that need to be simulated to perform the task.
Algorithm 3 Backward Forward Algorithm

\[\begin{align*}
\triangleright \text{With } [t_0, t_1] \text{ the interval of simulation for neuron } i \in I \\
1. \text{Initialize the family } V \text{ of non empty neighborhoods with } \{(i, [t_0, t_1])\} \\
2. \text{Initialize the family of points } P = \emptyset \\
\triangleright \text{Each point is a time } T \text{ with 3 marks: } j_T \text{ is the neuron on which } T \\
\text{appears, } V_T \text{ for the choice of neighborhood, } X_T \text{ for the thinning step (accepted/rejected)} \\
3. \text{Draw } E \text{ an exponential variable with parameter } M \\
4. \text{Schedule } T_{next} = t_0 + E \\
5. \text{while } T_{next} < t_1 \text{ do} \\
6. \text{Append to } P, \text{ the point } T_{next}, \text{ with 3 marks: } j_{T_{next}} = i, V_{T_{next}} = \text{n.a.} \\
\text{and } X_{T_{next}} = \text{n.a.} \text{ (n.a. stand for not assigned yet)} \\
7. \text{while } \text{There are points } T \text{ in } P \text{ with } V_T = \text{n.a.} \text{ do} \\
8. \text{for each point } T \text{ in } P \text{ with } V_T = \text{n.a.} \text{ do} \\
9. \text{Update } V_T \text{ by drawing } V_T \text{ according to } \lambda_{j_T} \text{ shifted at time } T. \\
10. \text{if } V_T \neq \emptyset \text{ then} \\
11. \quad \text{Find the portion of time/neurons in } V_T \text{ which does not intersect the existing non empty neighborhoods in } V \\
12. \quad \text{Simulate on it a Poisson process with rate } M \\
13. \quad \text{Append the simulated points, } T', \text{ if any, to } P \text{ with their neuron } j_{T'} \text{ and with } V_{T'} = X_{T'} = \text{n.a} \\
14. \quad \text{Append } V_T \text{ to } V \\
15. \text{end if} \\
16. \text{end for} \\
17. \text{end while} \\
18. \text{Sort the } T's \text{ in } P \text{ with } X_T = \text{n.a.} \text{ in increasing order} \\
19. \text{for each of them starting with the most backward do} \\
20. \quad \text{Draw } X_T \text{ as a Bernoulli variable with parameter } \frac{\lambda_{j_T}(T)}{M} \\
21. \text{end for} \\
22. \text{Draw } E' \text{ another exponential variable with parameter } M \\
23. \text{ } T_{next} \leftarrow T_{next} + E' \\
24. \text{end while} \\
25. \text{The desired points are the points in } P \text{ with marks } j_T = i, X_T = 1 \text{ and that} \\
\text{appear in } [t_0, t_1] \\
\triangleright \text{It is possible that the algorithm generated points before } t_0 \text{ and they have} \\
\text{to be removed}
\end{align*}\]

At the difference with Algorithm 2, the main point is that in the backward part we pick at random all the points that may influence the thinning step. The fact that this loop ends comes from the following Proposition.

Proposition 1 \text{If}

\[\sup_{\nu \in \mathcal{Y}} \sum_{i \in \mathbb{J}_i} \lambda_i(v) l(v) M < 1.\] \tag{3}

then the backward part of Algorithm 3 ends almost surely in finite time.

The proof is postponed to Appendix B. It is based on branching process arguments. Basically if in Steps 8–16, we produce in average less than one point, either because we picked the empty set in \(V_T\) or because the simulation of the Poisson process ended up with a small amount of points, eventually none, then the loop ends almost surely because there is an extinction of the corresponding branching process.

In the backward part, one of the most delicate parts consists in being sure that we add new points only if we have not visited this portion of time/neurons before (see Steps 11–13). If we do not make this verification, we may not have the same past depending on the neuron we are looking at and the procedure would not simulate the process we want.

In the forward part, because the backward algorithm stopped just before, we are first sure to have assess all \(V_T\)'s. Since \(\phi_{j_T}^E(t)\) is \(\mathcal{F}_E^{t}\) measurable for all \(t, \phi_{j_T}^E(T)\) only depends on the points in \(P\) with mark \(X_T = 1\) inside \(V_T\). The problem in Algorithm 2, phrased differently, is that we do not know the marks \(X_T\) of the previous points when we have to compute \(\phi_{j_T}^E(T)\). But in the forward part of Algorithm 3, we are sure that the most backward point for which the thinning \((X_T = \text{n.a.})\) has not taken place satisfies...
either $V_T = \emptyset$

- or $V_T \neq \emptyset$ but either there are no simulated points in the corresponding $V_T$ or the points there come from previous rounds of the loop (Step 5). Therefore, their marks $X_T$ have been assigned.

Therefore, with the Backward Forward algorithm, and at the difference to Algorithm 2, we take the points in an order for which we are sure that we know the previous needed marks.

Figure 1 describes an example to go step by step through Algorithm 3. The 

- backward steps determine all the points that may influence the acceptation/rejection of point $T_{next}$.
- forward steps finally select the points.

Illustration

To illustrate in practice the algorithm, we have simulated a Hawkes process as given in (1). Indeed such a process has a Kalikow decomposition (2) with bound $M$ and neighborhood family $\mathcal{V}$ constituted of the $v$’s of the form $v = \{(j, [-A, 0])\}$ for some neuron $j$ in $I$. To do that, we need the following choices:
\(\lambda_i(\emptyset) = 1 - \sum_{j \in I} w_{ij} \quad \text{and} \quad \phi_i^\emptyset = \frac{v_i}{\lambda_i(\emptyset)}\)

and for \(\nu\) of the form \(\nu = \{(j, [-A, 0])\}\) for some neuron \(j\) in \(I\),

\(\lambda_i(\nu) = w_{ij} \quad \text{and} \quad \phi_i^\nu = Nb_{[-A, 0]}^j \wedge M.\)

We have taken \(I = \mathbb{Z}^2\) and the \(w_{ij}\) proportional to a discretized centre symmetric bivariate Gaussian distribution of standard deviation \(\sigma\). More precisely, once \(\lambda_i(\emptyset) = \lambda_\emptyset\) fixed, picking according to \(\lambda_i\) consists in

- choosing whether \(V\) is empty or not with probability \(\lambda_\emptyset\)
- if \(V \neq \emptyset\), choosing \(V = \{(j, [-A, 0])\}\) with \(j - i = \text{round}(W)\) and \(W\) obeys a bivariate \(N(0, \sigma^2)\).

In all the sequel, the simulation is made for neuron \(i = (0, 0)\) with \(i_0 = 0, t_i = 100\) (see Algorithm 3). The parameters \(M, \lambda_\emptyset\) and \(\sigma\) vary. The parameters \(v_i = \nu\) and \(A\) are fixed accordingly by

\(v = 0.9M\lambda_\emptyset \quad \text{and} \quad A = 0.9M^{-1}(1 - \lambda_\emptyset)^{-1},\)

to ensure that \(\phi_i^\emptyset < M\) and (3), which amounts here to \((1 - \lambda_\emptyset)AM < 1.\)

In Fig. 2a, with \(M = 2, \sigma = 1\) and \(\lambda_\emptyset\) small, we see the overall spread of the algorithm around the neuron to simulate (here \((0, 0))\). Because we chose a Gaussian variable with small variance for the \(\lambda_i\)’s, the spread is weak and the neurons very close to the neuron to simulate are requested a lot of time at Steps 9–11 of the algorithm. This is also where the algorithm spent the biggest amount of time to simulate Poisson processes. Note also that roughly to simulate 80 points, we need to simulate 10 times more points globally in the infinite network. Remark also in Fig. 2b the avalanche phenomenon, typical of Hawkes processes: for instance the small cluster of black points on neuron with label 0 (i.e., \((0, 0)\)) around time 22, is likely to be due to an excitation coming for the spikes generated (and accepted) on neuron labeled 8 and self excitation. The beauty of the algorithm is that we do not need to have the whole time line of neuron 8 to trigger neuron 0, but only the small blue pieces: we just request them at random, depending on the Kalikow decomposition.

In Fig. 3, we can first observe that when the parameter which governs the range of \(\lambda_i\)’s increases, the global spread of the algorithm increases. In particular, comparing the top left of Figs. 3 to 2 where the only parameter that changes is \(\sigma\), we see that the algorithm is going much further away and simulates much more points for a sensible equal number of points to generate (and accept) on neuron \((0,0)\). Moreover, we can observe that

- From left to right, by increasing \(\lambda_\emptyset\), it is more likely to pick an empty neighborhood and as a consequence,
the spread of the algorithm is smaller. By increasing $v = 0.9M\lambda_p$, this also increases the total number of points produced on neuron (0;0).

From top to bottom, by increasing $M$, there are more points which are simulated in the Poisson processes (Step 12 of Algorithm 3) and there is also a stronger interaction (we do not truncate that much the number of points in $\phi^i$). Therefore, the spread becomes larger and more uniform too, because there are globally more points that are making requests. Moreover, by having a basic rate $M$ which is 10 times bigger, we have to simulate roughly 10 times more points.

**Conclusion**

We derived a new algorithm for simulating the behavior of one neuron embedded in an infinite network. This is possible thanks to the Kalikow decomposition which allows picking at random the influencing neurons. As seen in the last section, it is computationally tractable in practice to simulate open systems in the physical sense. A question that remains open for future work is whether we can prove that such a decomposition exists for a wide variety of processes, as it has been shown in discrete time (see [9, 10, 18]).

**Acknowledgements**

This work was supported by the French government, through the UCA*Investissements d’Avenir managed by the National Research Agency (ANR-15-IDEX-01) and by the interdisciplinary Institute for Modeling in Neuroscience and Cognition (NeuroMod) of the Université Côte d’Azur. The authors would like to thank Professor E. Löcherbach from Paris 1 for great discussions about Kalikow decomposition and Forward Backward Algorithm.
A Link Between Algorithm 2 and the Kalikow Decomposition

To prove that Algorithm 2 returns the desired processes, let us use some additional and more mathematical notation. Note that all the points simulated on neuron $i$ before being accepted or not can be seen as coming from a common Poisson process of intensity $M$, denoted $I_i$. For any $i \in I$, we denote the arrival times of $I_i$, $(T_n^{i})_{n \in \mathbb{Z}}$, with $T_1^i$ being the first positive time.

As in Step 6 of Algorithm 2, we attach to each point of $\Pi^i$ a stochastic mark $X$ given by

$$X_n^i = \begin{cases} 1 & \text{if } T_n^i \text{ is accepted in the thinning procedure} \\ 0 & \text{otherwise}. \end{cases}$$

Let us also define $V_n^i$, the neighborhood choice of $T_n^i$ picked at random and independently of anything else according to $\lambda_i$ and shifted at time $T_n^i$.

In addition, for any $i \in I$, define $N_n^i = (T_n^i, X_n^i)_{n \in \mathbb{Z}}$ an $E$-marked point process with $E = \{0; 1\}$. In particular, following the notation in Chapter VIII of [2], for any $i \in I$, let

$$N_n^i(\text{mark}) = \sum_{m \in \mathbb{Z}} 1_{X_m^i = \text{mark}} 1_{T_m^i \leq t} \quad \text{for } \text{mark} \in E$$

$$F^{N}(r) = \prod_{n=1}^{\infty} \sigma(N_n^i(0), N_n^i(1); s < t) \quad \text{and} \quad F^{N(1)}(r) = \prod_{n=1}^{\infty} \sigma(N_n^i(1); s < t).$$

Moreover note that $(N_n^i(1))_{n \in \mathbb{Z}}$ is the counting process associated with the point process $P$ simulated by Algorithm 2. Let us denote by $\varphi_i(t)$ the formula given by (2) and shifted at time $t$. Note that since the $\varphi_i$’s are $F^{N}(r)$ measurable, we have that $F^{N(1)}(r)$ measurable. We also denote $\varphi_i(t)$ the formula of $\varphi_i^r$ shifted at time $t$.

With this notation, we can prove the following.

**Proposition 2** The process $(N_n^i(1))_{n \in \mathbb{Z}}$ admits $\varphi_i(t)$ as $F^{N(1)}$-predictable intensity.

**Proof** Following the technique in Chapter 2 of [2], let us take $C_t$, a non-negative predictable function with respect to (w.r.t) $F^{N(1)}(r)$ that is $F^{N(1)}(r)$ measurable and therefore $F_{r-}$ measurable. We have, for any $i \in I$,

$$E \left( \int_0^\infty C_t dN^i_t(1) \right) = \sum_{n=1}^{\infty} E \left( C_{T_n^i} 1_{X_n^i = 1} \right)$$

Note that by Theorem T35 at Appendix A1 of [2], any point $T$ should be understood as a stopping time and that by Theorem T30 at Appendix A2 of [2],

$$F^{N(1)}_{r-} = \bigcap_{j} \sigma(T_m^j, X_m^j \text{ such that } T_m^j < T}$$

So

$$E \left( \int_0^\infty C_t dN^i_t(1) \right) = \sum_{n=1}^{\infty} E \left( C_{T_n^i} 1_{X_n^i = 1} \frac{F^{N(1)}_{T_n^i-}}{M} \right) = \sum_{n=1}^{\infty} E \left( C_{T_n^i} \frac{\varphi_i^r(T_n^i)}{M} \right).$$

Let us now integrate with respect to the choice $V_n^i$, which is independent of anything else.

$$E \left( \int_0^\infty C_t dN^i_t(1) \right) = \sum_{n=1}^{\infty} E \left( C_{T_n^i} \frac{\varphi_i^r(T_n^i)}{M} \right)$$

$$= \sum_{n=1}^{\infty} E \left( C_{T_n^i} \frac{\varphi_i^r(t)}{M} d\Pi^i(t) \right).$$

Since $\Pi^i$ is a Poisson process with respect to $(F^{N(1)}_r)$, with intensity $M$, and since $C_t \frac{\varphi_i^r(t)}{M}$ is $F_{r-}$ measurable, we finally have that

$$E \left( \int_0^\infty C_t dN^i_t(1) \right) = E \left( \int_0^\infty C_t \varphi_i(t) dt \right),$$

which ends the proof.

**B Proof of Proposition 1**

**Proof** We do the proof for the backward part, starting with $T = T_{next}$ as the next point after $t_0$ (Step 4 of Algorithm 3), the proof being similar for the other $T_{next}$ generated at Step 23. We construct a tree with root $(i, T)$. For each point $(j, T')$ in the tree, the points which are simulated in $V_{T'}$ (Step 12 of Algorithm 3) define the children of $(j, T')$ in the tree. This forms the tree $\mathcal{F}$.

Let us now build a tree $\mathcal{F}$ with root $(i, T)$ (that includes the previous tree) by mimicking the previous procedure in the backward part, except that we simulate on the whole neighborhood even if it has a part that intersects with previous neighborhoods (if they exist) (Steps 11–12 of Algorithm 3). By doing so, we make the number of children at each node independent of anything else.
If the tree $\mathcal{C}$ goes extinct then so does the tree $\mathcal{T}$ and the backward part of the algorithm terminates.

But if one only counts the number of children in the tree $\mathcal{C}$, we have a marked branching process whose reproduction distribution for the mark $i$ is given by

- no children with probability $\lambda_i(\emptyset)$
- Poissonian number of children with parameter $l(v)M$ if $v$ is the chosen neighborhood with probability $\lambda_i(v)$

This gives that the average number of children issued from a node with the mark $i$ is

$$\zeta_i = \lambda_i(\emptyset) \times 0 + \sum_{v \in \mathcal{V}, v \neq \emptyset} \lambda_i(v)(v)M.$$ 

If we denote $\mathcal{C}^k$ as the collection of points in the tree $\mathcal{C}$ at generation $k$, and by $K_T$ the set of points generated independently as a Poisson process of rate $M$ inside $V_T$, we see recursively that

$$\mathcal{C}^{k+1} = \bigcup_{T' \in \mathcal{C}^k} K_{T'}.$$ 

But

$$\mathbb{E}(|K_T||T') = \zeta_{j_{T'}}.$$ 

Therefore, if we denote the total number of sites in $\mathcal{C}^k$ by $Z^{(k)}$, we have

$$\mathbb{E}(Z^{(k+1)}|\mathcal{C}^k) \leq Z^{(k)} \sup_{i \in I} \zeta_i.$$ 

One can then conclude by recursion that

$$\mathbb{E}(Z^{(k)}) \leq (\sup_{i \in I} \zeta_i)^k < 1.$$ 

The last inequality uses the sparsity neighborhood assumption. Then, we deduce that the mean number of children in each generation goes to 0 as $k$ tends to infinity. So using classical branching techniques in [15], we conclude that the tree $\mathcal{C}$ will go extinct almost surely. This also implies that the backward steps end a.s. \(\square\)

References

1. Andersen PK, Borgan O, Gill R, Keiding N. Statistical models based on counting processes. Berlin: Springer; 1996.
2. Brémaud P. Point processes and queues: martingale dynamics. Berlin: Springer; 1981.
3. Brémaud P, Massoulié L. Stability of nonlinear Hawkes processes. Ann Probab. 1996;24:1563–88.
4. Comets F, Fernandez R, Ferrari PA. Processes with long memory: regenerative construction and perfect simulation. Ann Appl Probab. 2002;3:921–43.
5. Muzy A. Exploiting activity for the modeling and simulation of dynamics and learning processes in hierarchical (neurocognitive) systems. Mag Comput Sci Eng. 2019;21:83–93.
6. Dassios A, Zhao H. Exact simulation of Hawkes process with exponentially decaying intensity. Electron Commun Probab. 2013;18(62):1–13.
7. Didelez V. Graphical models of mark processes based on local independence. J R Stat Soc B. 2008;70(1):245–64.
8. Fernández R, Ferrari P, Galves A. Coupling, renewal and perfect simulation of chains of infinite order. In: Notes of a course in the V Brazilian School of Probability, Ubatuba, July 30–August 4, 2001. http://www.staff.science.uu.nl/~ferna107/.
9. Galves A, Löcherbach E. Infinite systems of interacting chains with memory of variable length—a stochastic model for biological neural nets. J Stat Phys. 2013;151(5):896–921.
10. Galves A, Löcherbach E. Modeling networks of spiking neurons as interacting processes with memory of variable length. Journal de la Société Française de Statistiques. 2016;157:17–32.
11. Hodara P, Löcherbach E. Hawkes Processes with variable length memory and an infinite number of components, Adv. Appl. Probab. 2017;49: 84–107 title = Hawkes Processes with variable length memory and an infinite number of components, year = 2016.
12. Kalikow S. Random markov processes and uniform martingales. Isr J Math. 1990;71(1):33–54.
13. Lewis PAW, Shedler GS. Simulation of nonhomogeneous Poisson processes. Monterey, California: Naval Postgraduate School; 1978.
14. Mascart C, Muzy A, Reynaud-Bouret P. Centralized and distributed simulations of point processes using local independence graphs: A computational complexity analysis, under finalization for Arxiv deposit 2019.
15. Méleard S, Aléatoire: introduction à la théorie et au calcul des probabilités, Editions de l’écalle polytechnique, 2010, pp. 185–194.
16. Möller J, Rasmussen JG. Perfect simulation of Hawkes processes. Adv Appl Probab. 2005;37:629–46.
17. Ogata Y. On Lewis’ simulation method for point processes. IEEE Trans Inf Theory. 1981;27:23–31.
18. Ost G, Reynaud-Bouret P. Sparse space-time models: concentration inequalities and Lasso. Under revision. https://arxiv.org/abs/1807.07615.
19. Peters EAJF, de With G. Rejection-free Monte Carlo sampling for general potentials. Phys Rev E. 2012;85:026703.
20. Reynaud-Bouret P, Schbath S. Adaptive estimation for Hawkes processes: application to genome analysis. Ann Stat. 2010;38(5):2781–822.
21. Tocher KD. PLUS/GPS III Specification, United Steel Companies Ltd, Department of Operational Research. 1967.
22. Vere-Jones D, Ozaki T. Some examples of statistical estimation applied to earthquake data. Ann Inst Stat Math. 1982;34(B):189–207.
23. Zeigler BP. Theory of modelling and simulation. New York: Wiley-Interscience Publication; 1976.
24. Zeigler BP, Muzy A, Kofman E. Theory of modeling and simulation: discrete event & iterative system computational foundations. New York: Academic Press; 2018.