Networks and Random Processes

Hand-out 3

Random sequential update, Gillespie algorithm

The properties of Poisson processes (see hand-out 2) can be used to set up efficient sampling algorithms for stochastic particle systems. Here we focus on a system with state space $S = \{0,1\}^\Lambda$ with lattice Λ and flip dynamics (for example the contact process), and sketch two algorithms: the random sequential update, and the rejection-free Gillespie algorithm.

Random sequential update.

Pro: works with fixed sampling rates and probabilities that can be pre-computed and stored, computationally cheap and simple to implement
Con: if transition rates are heterogeneous, proposed updates may be rejected with high probability which leads to oversampling and waste of computational time
$\to$ works best for processes with homogeneous, similar transition rates.

To resolve the full dynamics on site $x \in \Lambda$, the (fixed) sampling rate should be $r_x = \max_{\eta\in S} c(\eta,\eta^x)$ determined by the fastest process. The independent PPs on each site add up, and the next possible event in the whole system is sampled at rate $R = \sum_{x\in\Lambda} r_x$. By the thinning property, the probability that it happens on site x is given by $p_x = r_x/R$. This leads to the following algorithm:
Pick η_0 from the initial distribution and set $t = 0$. Then repeat iteratively:

(1) update the time counter by $t+ = \mathrm{Exp}(R)$,
(2) pick a site x with probability p_x,
(3) update (flip) site x with probability $c(\eta,\eta^x)/r_x$.

So in total, the process $\eta \to \eta^x$ happens with the correct rate $\quad R\,\frac{r_x}{R}\,\frac{c(\eta,\eta^x)}{r_x} = c(\eta,\eta^x)$.
For example, for the 1D contact process on $\Lambda = \{1,\dots,L\}$ with periodic boundaries and rates

$$c(\eta,\eta^x) = \eta(x) + \lambda\big(1-\eta(x)\big)\big(\eta(x-1)+\eta(x+1)\big)$$

we have $r_x = r = \max\{1, 2\lambda\}$, and thus $p_x = 1/L$ choosing sites uniformly and $R = rL$.

Gillespie algorithm.

Pro: is rejection-free, i.e. a transition occurs in every step
Con: can be computationally heavy since rates and probabilities are computed in each step
$\to$ works best for processes with very heterogeneous transition rates.

The sampling rate $R(\eta) = \sum_{x\in\Lambda} c(\eta,\eta^x)$ is state-dependent and needs updating in each time step.
Pick η_0 from the initial distribution and set $t = 0$. Then repeat iteratively:

(0) compute/update the sampling rate $R(\eta)$,
(1) update the time counter by $t+ = \mathrm{Exp}(R(\eta))$,
(2) pick a site x with probability $p_x(\eta) = c(\eta,\eta^x)/R(\eta)$,
(3) update (flip) site x.

So in total, the process $\eta \to \eta^x$ happens with the correct rate $\quad R(\eta)\,\frac{c(\eta,\eta^x)}{R(\eta)} = c(\eta,\eta^x)$.

Simplified time counter.
In both algorithms, $R = O(L)$ is of order of the system size, so the increments $\tau_i \sim \mathrm{Exp}(R)$ of the time counter are of order $1/L$. By the **scaling property** $\alpha\mathrm{Exp}(\beta) \sim \mathrm{Exp}(\beta/\alpha)$ of exponential rv's (check!), we have

$$\tau_i \sim Exp(R) \sim \frac{1}{R}\,\tilde{\tau}_i \quad \text{with normalized} \quad \tilde{\tau}_i \sim Exp(1)\ .$$

To simulate up to a time $T = O(1)$ we therefore need of order $RT = O(L)$ sampling increments τ_i. The time counter of the simulation is then

$$t = \sum_{i=1}^{RT} \tau_i = \frac{1}{R} \sum_{i=1}^{RT} \tilde{\tau}_i = T + O(L^{-1/2}) \to T \quad \text{as } L \to \infty\ ,$$

by the law of large numbers. So if we just replace the increments τ_i by their mean $1/R$, i.e. use

(1)' update the time counter by $t+ = 1/R$

instead of the computationally more expensive (1), the error in t is of order $L^{-1/2}$ by the central limit theorem. This is often negligible for large L unless one is interested in very precise time statistics.