# Advanced topics in Markov-chain Monte Carlo

## Lecture 4: Complements on non-reversible hard-sphere algorithms Perfect sampling in Markov-chain Monte Carlo

Werner Krauth

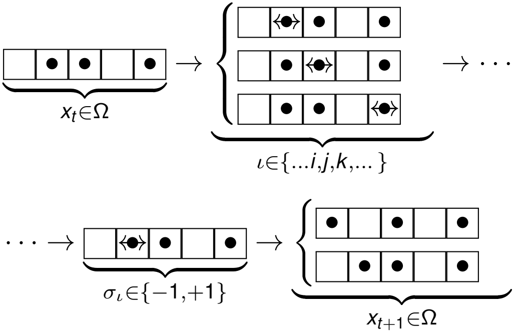
ICFP -Master Course Ecole Normale Supérieure, Paris, France

08 February 2023

# Complement: non-reversible hard-sphere algorithms

- Metropolis algorithm $\to$ Symmetric simple exclusion process
- Forward algorithm $\to$ Totally asymmetric simple exclusion process (TASEP)
- Lifted forward algorithm $\to$ Lifted TASEP

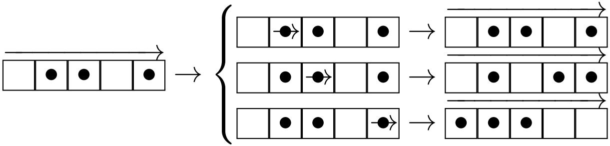
# Symmetric simple exclusion process (SSEP)



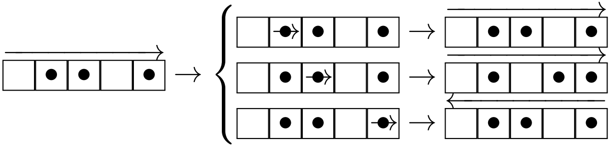

- here: periodic lattice
- satisfies detailed balance with equal probability for each configuration $\{\ldots i, j, k, \ldots\}$

# Totally asymmetric simple exclusion process (TASEP)

On a periodic lattice:

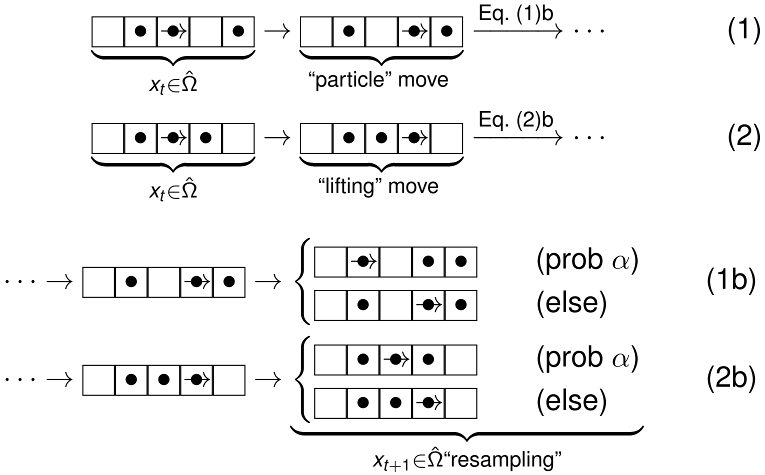


On a non-periodic lattice:



- satisfies global balance with equal probability for each lifted configuration $\{\ldots i, j, k, \ldots, \sigma\}$ with $\sigma \in \{\leftarrow, \rightarrow\}$
- is (half) a lifting of the SSEP

# Lifted TASEP




- Satisfies global balance with equal probability for each lifted configuration $\{\ldots i, \vec{j}, k, \ldots, \sigma\}$

# Perfect sampling - References

- J. G. Propp, D. B. Wilson **Exact sampling with coupled Markov chains and applications to statistical mechanics** `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.27.1022`
- W. Krauth "**Statistical Mechanics: Algorithms and Computations**" (Oxford University Press, 2006)

# Perfect sampling

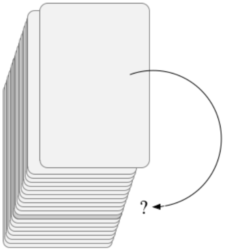
- Irreducible aperiodic (finite) Markov chains converge exponentially:

$$\max_{x \in \Omega} ||P^t(x,.) - \pi||_{\text{TV}} < C\alpha^t$$

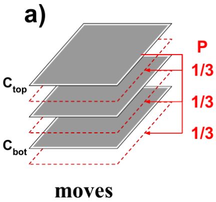
with $C > 0$ and $\alpha \in (0, 1)$.

- Exponential convergence $\implies$ time scale $1/\log\alpha$
- Normally $P^t \neq \pi$ for finite $t$.
- We now show in two examples that MCMC can often sample $\pi$ exactly.

# Shuffling of cards 1/5

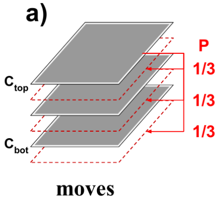



- $\Omega_N^{\text{shuffle}} = \{\text{Permutations of } \{1, \ldots, N\}\}$
- For $N = 3$:
  $\Omega_3^{\text{shuffle}} = \{1 \equiv \{1,2,3\}, 2 \equiv \{1,3,2\}, 3 \equiv \{2,1,3\}, 4 \equiv \{2,3,1\}, 5 \equiv \{3,1,2\}, 6 \equiv \{3,2,1\}\}$.
- $\pi^{t=0} = \delta((1, , \ldots, N))$




- $\Omega_3^{\text{shuffle}} = \{1 \equiv \{1,2,3\}, 2 \equiv \{1,3,2\}, 3 \equiv \{2,1,3\}, 4 \equiv \{2,3,1\}, 5 \equiv \{3,1,2\}, 6 \equiv \{3,2,1\}\}$.
-
$$P = \frac{1}{3}\begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

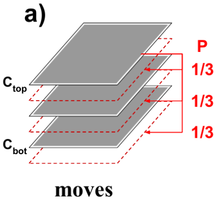
# Shuffling of cards 3/5




**procedure** top-to-random
**input** $\{c_1, \ldots, c_n\}$
$i \leftarrow \texttt{choice}(\{1, \ldots, n\})$
$\{\hat{c}_1, \ldots, \hat{c}_n\} \leftarrow \{c_2, \ldots, c_i, c_1, c_{i+1}, \ldots, c_n\}$
**output** $\{\hat{c}_1, \ldots, \hat{c}_n\}$

- Insert upper card ($c_1$) after card $i$ and before card $i + 1$
- NB: if $i = 1$, put it back on top.

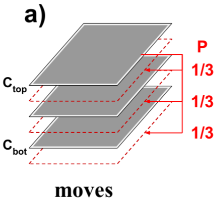
# Shuffling of cards 4/5




$$
\begin{array}{l}
\textbf{procedure } \texttt{top2random-stop} \\
\textbf{input } \{c_1, \ldots, c_n\} \\
c_{\text{first-n}} \leftarrow c_{\text{n}} \\
\textbf{for } t = 1, 2, \ldots \textbf{ do} \\
\left\{ \begin{array}{l}
\tilde{c}_1 \leftarrow c_1 \\
\{c_1, \ldots, c_n\} \leftarrow \texttt{top2random}(\{c_1, \ldots, c_n\}) \\
\textbf{if } (\tilde{c}_1 = c_{\text{first-n}}) \ \textbf{break}
\end{array} \right. \\
\textbf{output } \{c_1, \ldots, c_n, t\}
\end{array}
$$

- Expected running time: $n \log n$.
- Closely related to $\mathcal{O}(n)$ algorithm for random permutations.

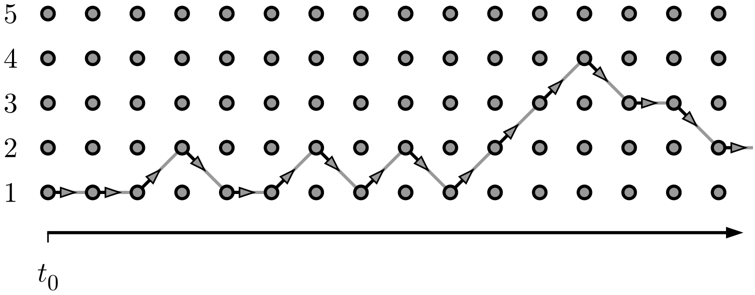
# Shuffling of cards 5/5




**procedure** direct-shuffle
**input** $\{c_1, \ldots, c_n\}$
**for** $t = 1, \ldots, n$ **do**
$$\left\{ \begin{array}{l} i \leftarrow \texttt{choice}(\{n-t+1, \ldots, n\}) \\ \{c_1, \ldots, c_n\} \leftarrow \{c_2, \ldots, c_i, c_1, c_{i+1}, \ldots, c_n\} \end{array} \right.$$
**output** $\{c_1, \ldots, c_n\}$

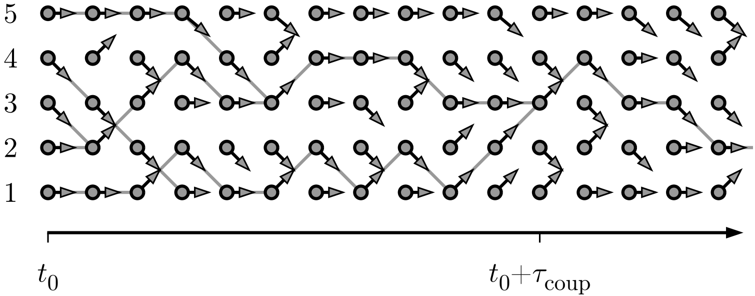
- Running time: $n$.
- Standard algorithm for generating random permutations.

# Markov chain (traditional view)




- Configuration $c_t$, move $\delta_t$.
- Set $t_0 = 0$.

# Markov chain (random maps), coupling 1/4



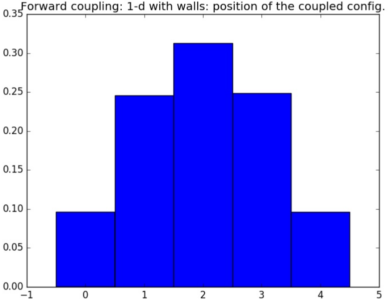

- Each configuration has its move at each time step.
- Coupling (Doeblin, 1930s).

# Markov chain (random maps), coupling 2/4

```
pos=[]
for stat in range(10000):
    posit=set(range(N))
    for t in range(1000000):
        posit = set([min(max(b + random.randint(-1, 1), 0), N - 1) for b in posit])
        if len(posit) == 1: break
    pos.append(posit.pop())
```

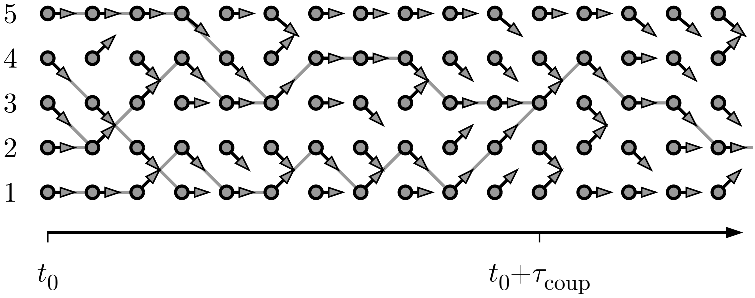
- Position of coupling not uniform.
- Coupling time larger than mixing time.

# Markov chain (random maps), coupling 3/4

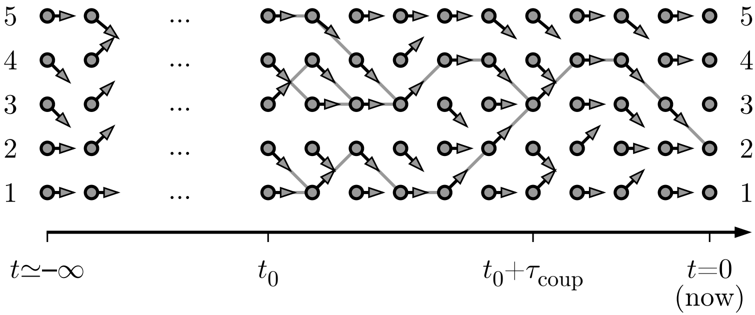



- Histogram of coupling position.

# Markov chain (random maps), coupling 4/4




- Each configuration has its move at each time step.
- Coupling (Doeblin, 1930s).



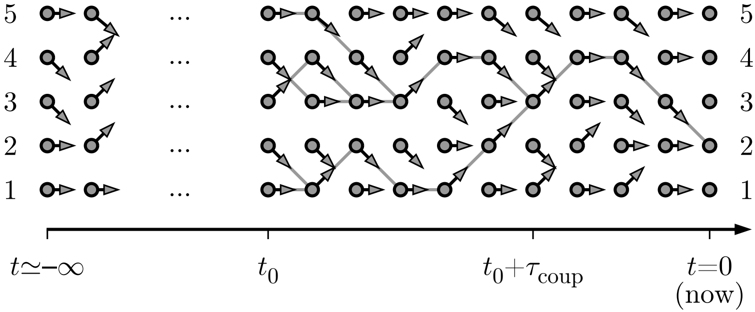

- Starting an MCMC simulation at $t = -\infty$
- Propp & Wilson (1997)

# Coupling from the past 2/8

```
pos = []
for statistic in range(100000):
   all_arrows = {}
   time_tot = 0
   while True:
      time_tot -= 1
      arrows = [random.randint(-1, 1) for i in range(N)]
      if arrows[0] == -1: arrows[0] = 0
      if arrows[N - 1] == 1: arrows[N - 1] = 0
      all_arrows[time_tot]=arrows
      positions=set(range(0, N))
      for t in range(time_tot, 0):
         positions = set([b + all_arrows[t][b] for b in positions])
         if len(positions) == 1: break
      if len(positions) == 1: break
```

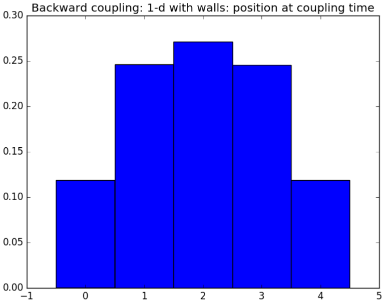
- Starting an MCMC simulation at $t = -\infty$
- Propp & Wilson (1997)

# Coupling from the past 3/8




- Starting an MCMC simulation at $t = -\infty$
- Propp & Wilson (1997)

# Coupling from the past 4/8



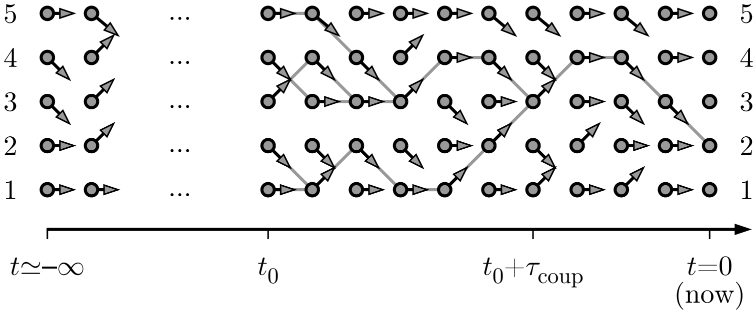

- Coupling position (in the past) non-uniform)

# Coupling from the past 5/8

```
for statistic in range(10000):
    all_arrows = {}
    time_tot = 0
    while True:
        time_tot -= 1
        old_pos = set(range(0, N))
        arrows = [random.randint(-1, 1) for i in range(N)]
        if arrows[0] == -1: arrows[0] = 0
        if arrows[N - 1] == 1: arrows[N - 1] = 0
        all_arrows[time_tot] = arrows
        positions = set(range(N))
        for t in range(time_tot, 0):
            positions = set([b + all_arrows[t][b] for b in positions])
        if len(positions) == 1: break
    a=positions.pop()
    pos.append(a)
```

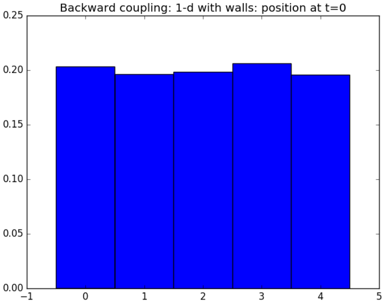
- Dictionary of random maps going back in time.

# Coupling from the past 6/8



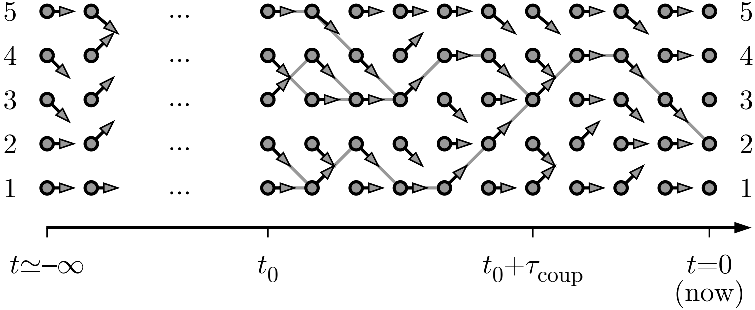

- Starting an MCMC simulation at $t = -\infty$
- Propp & Wilson (1997)

# Coupling from the past 7/8




- Perfect sample at $t = 0$, starting from $t = -\infty$
- Propp & Wilson (1997)

# Coupling from the past 8/8




- Try it yourself!