Perfect sampling using dynamic programming

Séminaire Équipe CALIN

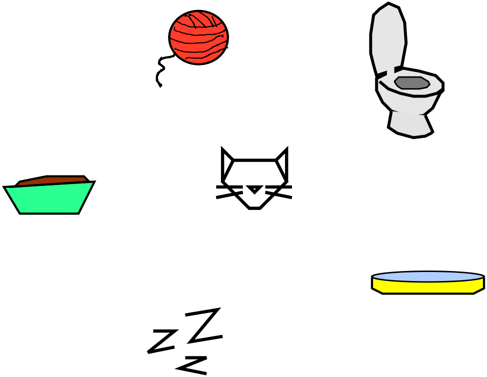
Christelle Rovetta

Équipe AMIB(io), Laboratoire LIX - Inria Saclay - LRI - RNALand

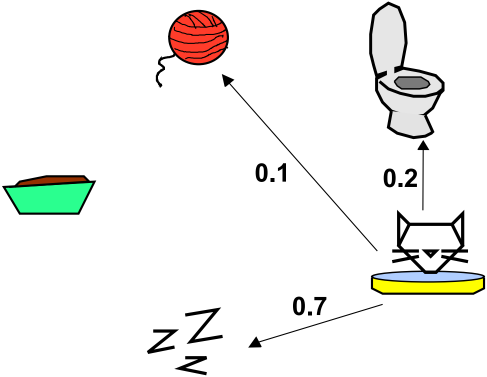
17th April 2018

What is Athena doing?

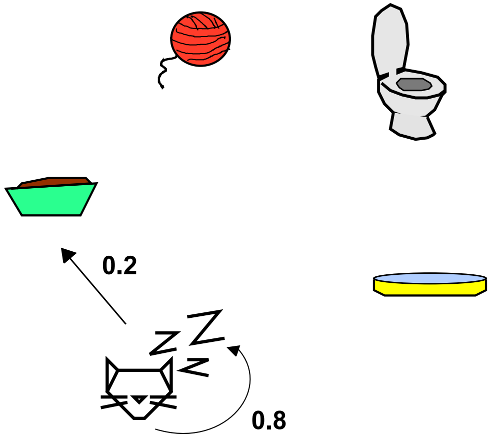
State space



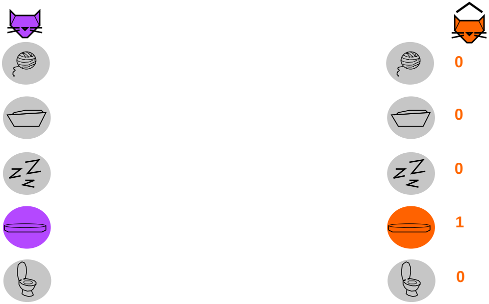
State space



State space

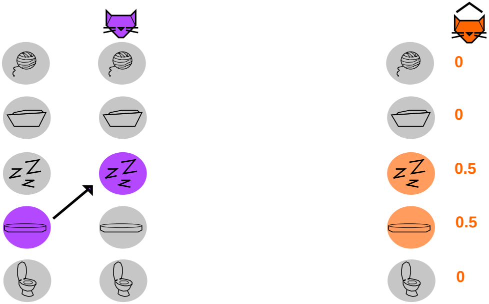


Markov chain



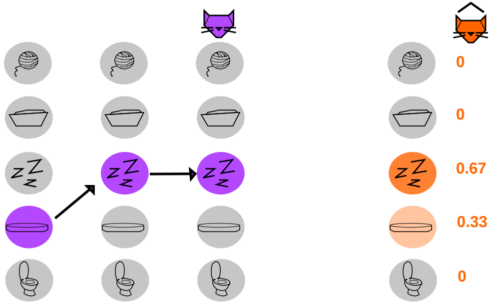
- Arithmetic mean $\widehat{\pi}$

Markov chain



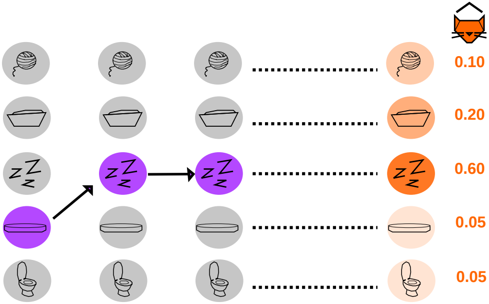
- Arithmetic mean $\widehat{\pi}$

Markov chain



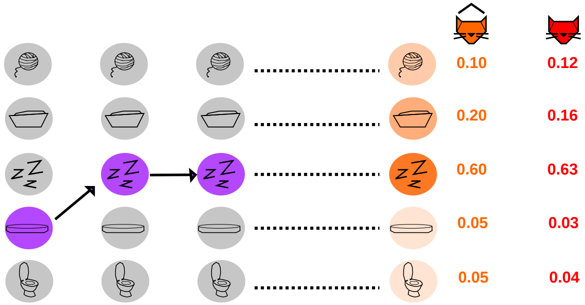
- Arithmetic mean $\widehat{\pi}$

Markov chain



- Arithmetic mean $\widehat{\pi}$

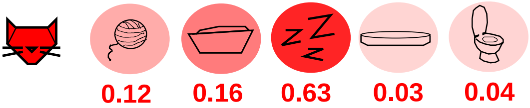
Markov chain



- Arithmetic mean $\widehat{\pi}$
- Stationary distribution π (solution of $\pi P = \pi$)

Sample the stationary distribution

- State space: $|\mathcal{S}| < \infty$
- Ergodic Markov chain $(X_n)_{n \in Z}$ on $\mathcal{S}$
- Stationary distribution π



- **Sample a random object according** π
- Very large $\mathcal{S}$

Markov Chain Monte Carlo (MCMC)

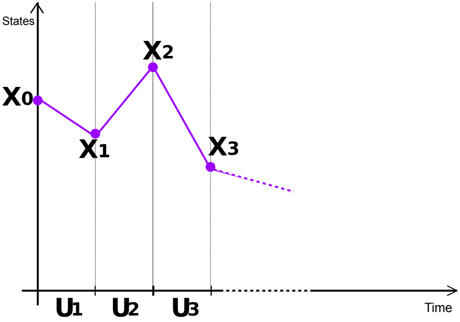
Markov chain convergence theorem

For all initial distributions $X_n \sim \pi$ when $n \to \infty$

Simulate the Markov chain

- $(U_n)_{n \in Z}$ an i.i.d sequence of random variables

$$\begin{cases} X_0 = x \in \mathcal{S} \\ X_{n+1} = \mathrm{update}(X_n, U_{n+1}) \end{cases}$$



Markov Chain Monte Carlo (MCMC)

Markov chain convergence theorem

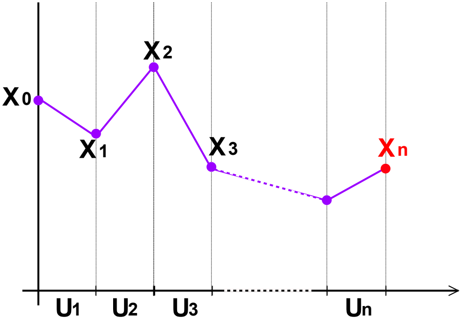
For all initial distributions $X_n \sim \pi$ when $n \to \infty$

Simulate the Markov chain

- $(U_n)_{n \in Z}$ an i.i.d sequence of random variables

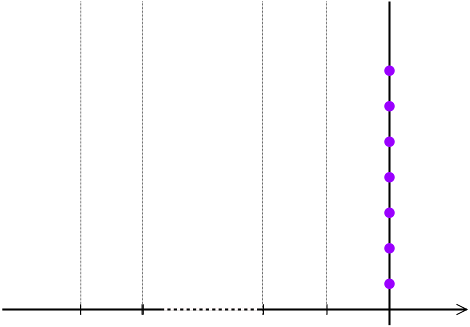
$$\begin{cases} X_0 = x \in \mathcal{S} \\ X_{n+1} = \mathrm{update}(X_n, U_{n+1}) \end{cases}$$

- How to detect the stopping criterion?



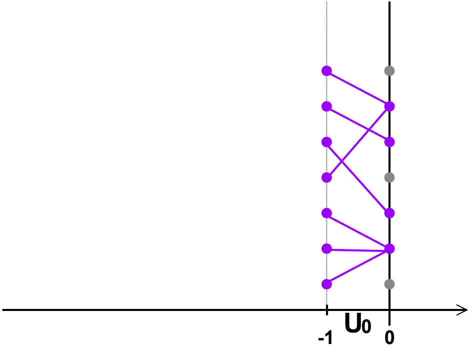
Perfect sampling algorithm

- Perfect sampling algorithm [Propp – Wilson, 1996]
  - Produces $y \sim \pi$
  - Stopping criterion automatically detected
  - Uses coupling from the past



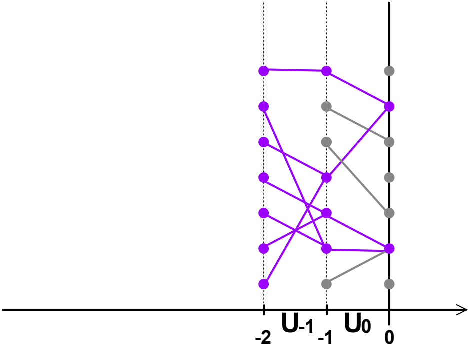
Perfect sampling algorithm

- Perfect sampling algorithm [Propp – Wilson, 1996]
  - Produces $y \sim \pi$
  - Stopping criterion automatically detected
  - Uses coupling from the past



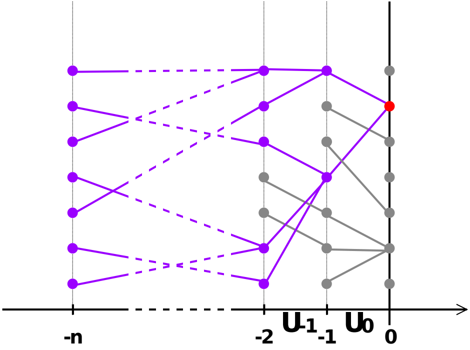
Perfect sampling algorithm

- Perfect sampling algorithm [Propp – Wilson, 1996]
  - Produces $y \sim \pi$
  - Stopping criterion automatically detected
  - Uses coupling from the past



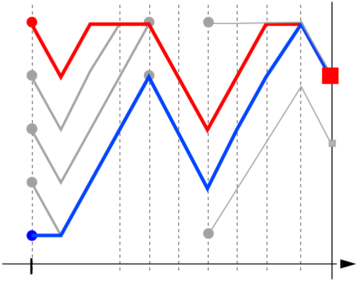
Perfect sampling algorithm

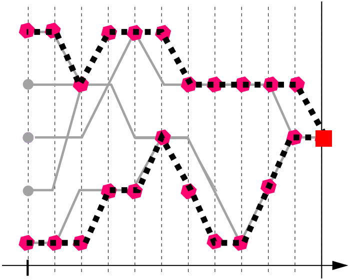
- Perfect sampling algorithm [Propp – Wilson, 1996]
  - Produces $y \sim \pi$
  - Stopping criterion automatically detected
  - Uses coupling from the past
- Starts from all states, complexity at least in $O(|\mathcal{S}|)$



Perfect sampling algorithm

- Perfect sampling algorithm [Propp – Wilson, 1996]
  - Produces $y \sim \pi$
  - Stopping criterion automatically detected
  - Uses coupling from the past
- Starts from all states, complexity at least in $O(|\mathcal{S}|)$
- Find strategies (monotone chains, envelopes,...)





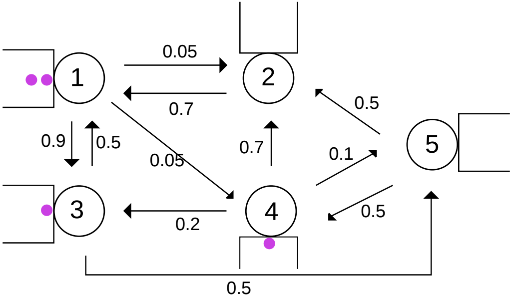
Queueing networks

- Introduced by Erlang in 1917 to describe the Copenhagen telephone exchange
- Queues are everywhere in computing systems
- Analyze various kinds of system performance (average waiting time, expected number of customers waiting, ...)

- Usually modeled by an ergodic Markov chain
- Computer simulation

Closed queueing network

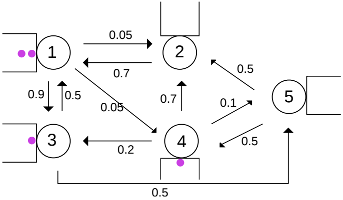
Customers are not allowed leave the network



- $K = 5$ queues, $M = 4$ customers
- State: $\mathbf{x} = (2, 0, 1, 1, 0)$
- Sample π

Product form

- Gordon-Newell networks



Gordon-Newell theorem

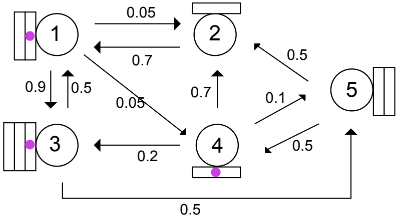
$$\pi_{\mathbf{x}} = \frac{1}{G(K,M)} \prod_{k \in \mathcal{Q}} \rho_k^{x_k} \quad \text{with} \quad G(K,M) = \sum_{\mathbf{x} \in \mathcal{S}} \prod_{k \in \mathcal{Q}} \rho_k^{x_k}.$$

- $G(K,M)$: normalization constant (partition function)
- Compute $G(K,M)$ in $O(KM)$, dynamic programming [Buzen '73]

Closed queueing network (monoclass)

- K queues $./M/1$ (exponential service rate)
- Finite capacity C_k in each queue
- Blocking policy: Repetitive service - random destination
- Strongly connected network

Example



- $K = 5$ queues, $M = 3$ customers, capacity $C = (2, 1, 3, 1, 2)$
- $\mathbf{x} = (1, 0, 1, 1, 0)$

State space

- K queues, M customers, capacity $C = (C_1, \ldots, C_K)$
- State space:

$$\mathcal{S} = \{\mathbf{x} \in \mathbb{N}^K \mid \sum_{k=1}^{K} x_k = M,\ \forall k\ 0 \leq x_k \leq C_k\}$$

- Number of states ($M \gg K$):

$$|\mathcal{S}| \leq \binom{M+K-1}{M} = \binom{M+K-1}{K-1} \text{ in } O\left(M^{K-1}\right)$$

Transition

- **Transition on a state**:

$$t_{i,j}(\mathbf{x}) = \begin{cases} \mathbf{x} - e_i + e_j & \text{if } x_i > 0 \text{ and } x_j < C_j, \\ \mathbf{x} & \text{otherwise } (x_i = 0 \text{ or } x_j = C_j), \end{cases}$$

$$\text{where } e_i \in \{0,1\}^K \qquad e_i(k) = \begin{cases} 1 & \text{if } i = k, \\ 0 & \text{otherwise.} \end{cases}$$

- **Transition on a set of states**:

$$t(S) := \bigcup_{\mathbf{x} \in S} t(\mathbf{x})$$

Markov chain modeling

- $(U_n)_{n\in Z} := (i_n, j_n)_{n\in Z}$ an i.i.d sequence of random variables
- System described by an ergodic Markov chain:

$$\begin{cases} X_0 \in \mathcal{S} \\ X_{n+1} = t_{U_{n+1}}(X_n) \end{cases}$$

- Unique stationary distribution π that is unknown
- GOAL: sample π with the perfect sampling algorithm

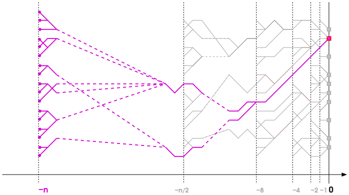
Perfect sampling algorithm

Perfect Sampling with States (PSS)

1. $n \leftarrow 1$
2. $t \leftarrow t_{U_{-1}}$
3. While $|t(\mathcal{S})| \neq 1$
4. $\quad n \leftarrow 2n$
5. $\quad t \leftarrow t_{U_{-1}} \circ \ldots \circ t_{U_{-n}}$
6. Return $t(\mathcal{S})$

- PROBLEM: $|\mathcal{S}|$ in $O(M^{K-1})$
- Find a strategy !

- Perfect sampling



A new stategy

Difficulty to adapt known stategies

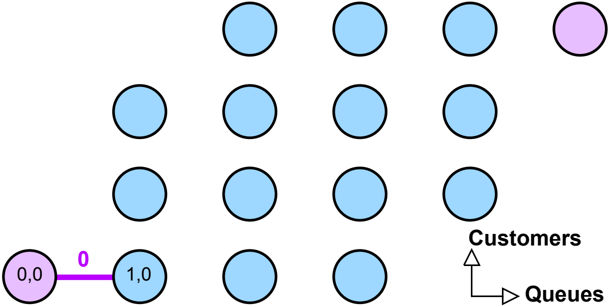
- Fixed number of customers ($\sum_{k=1}^{K} x_k = M$)
- No lattice structure

More structured representation of the state space

- **Reduce complexity:** $O(M^{K-1})$ **to** $O(KM^2)$
- Represent states as paths in a graph
- Realize transitions directly on the graph

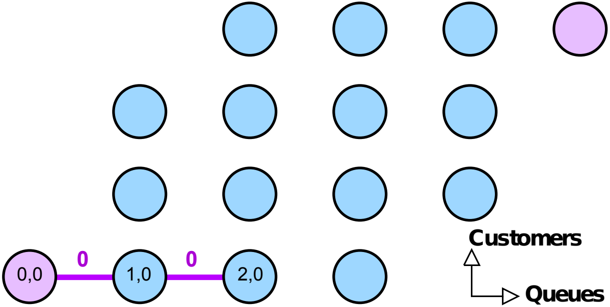
Diagram

- State:
  - $x = (0, 0, 2, 0, 1)$
- 5 queues, 3 customers, capacity $\mathbf{C} = (2, 1, 3, 1, 2)$
  - $\sum_{k=1}^{5} x_k = 3$
  - $\forall k \ 0 \leq x_k \leq C_k$
- Diagram



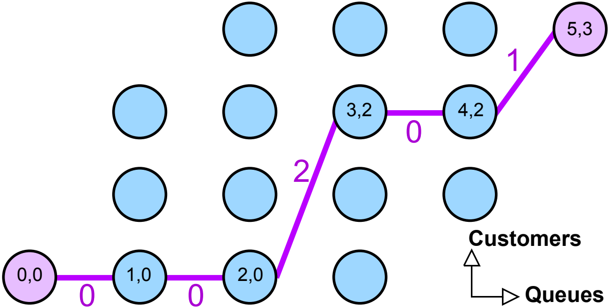
Diagram

- State:
  - $x = (0, 0, 2, 0, 1)$
- 5 queues, 3 customers, capacity $\mathbf{C} = (2, 1, 3, 1, 2)$
  - $\sum_{k=1}^{5} x_k = 3$
  - $\forall k \; 0 \leq x_k \leq C_k$
- Diagram



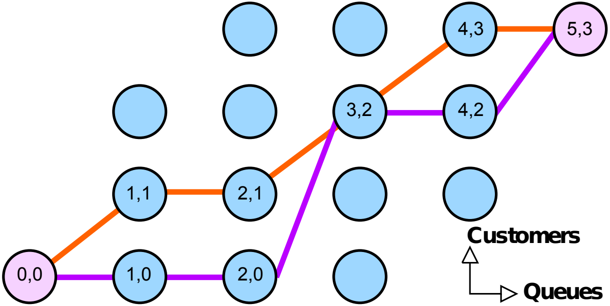
Diagram

- State:
  - $x = (0, 0, 2, 0, 1)$
- 5 queues, 3 customers, capacity $\mathbf{C} = (2, 1, 3, 1, 2)$
  - $\sum_{k=1}^{5} x_k = 3$
  - $\forall k \; 0 \leq x_k \leq C_k$
- Diagram



Diagram

- State:
  - $x = (0, 0, 2, 0, 1)$
  - $y = (1, 0, 1, 1, 0)$
- 5 queues, 3 customers, capacity $\mathbf{C} = (2, 1, 3, 1, 2)$
  - $\sum_{k=1}^{5} x_k = 3$
  - $\forall k \; 0 \leq x_k \leq C_k$
- Diagram

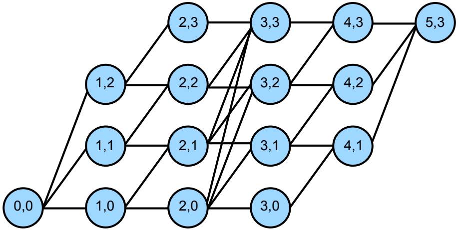


Diagram

- A **diagram** is a graph that encode a set of states
- A diagram is **complete** if it encodes all the states
- Number of arcs in a diagram: $O(KM^2)$

Example

$K = 5$ queues	5 columns of arcs
$M = 3$ customers	4 rows
$C = (2, 1, 3, 1, 2)$	$0 \leq \|\text{slopes}\| \leq 3$



States to Diagram: function ϕ

S

$(0, 0, 2, 0, 1)$
$(1, 0, 1, 1, 0)$

Diagram $\phi(S)$

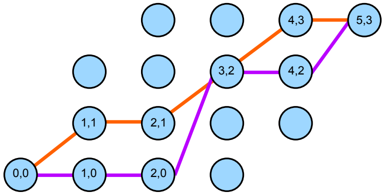
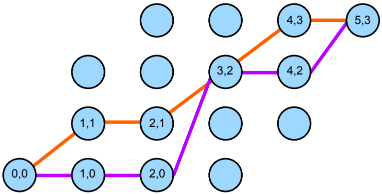


Diagram to states: function ψ

Diagram D



$\psi(D)$

$(0, 0, 2, 0, 1)$
$(1, 0, 1, 1, 0)$
$(0, 0, 2, 1, 0)$
$(1, 0, 1, 0, 1)$

Transformation function

$$
\begin{tikzcd}
S \arrow[d, phantom, sloped, "\subseteq"] \arrow[dr, "\phi"] & \\
\psi(D) \arrow[r, shift left, "\phi"] & D \arrow[l, shift left, "\psi"]
\end{tikzcd}
$$

Transformation function

- Galois connexion

$$
\begin{tikzcd}
S \arrow[d, phantom, sloped, "\subseteq"] \arrow[dr, "\phi"] & \\
\psi(D) \arrow[r, shift left, "\phi"] & D \arrow[l, shift left, "\psi"]
\end{tikzcd}
$$

Transition on a diagram

- Transition on a diagram:

$$T_{i,j} = \phi \circ t_{i,j} \circ \psi$$

- Good properties for perfect sampling
  - Preserves inclusion

$$S \subseteq \psi(D) \implies t_{i,j}(S) \subseteq \psi(T_{i,j}(D))$$

  - Preserves coupling

$$|\psi(D)| = 1 \implies |\psi(T_{i,j}(D))| = 1$$

- Efficient algorithm to compute transitions $T_{i,j}$ in $O(KM^2)$

Transition on a set of states

- Parameters: $K = 5$ queues, $M = 3$ customers, capacity $C = (2, 1, 3, 1, 2)$.

  $S = \{(0, 1, 1, 0, 0), (0, 1, 1, 1, 0), (0, 1, 0, 0, 2), (1, 0, 1, 1, 0)\} \subseteq \mathcal{S}$

- Transition $t_{4,2}(S)$?

Transition $t_{4,2}$ on S

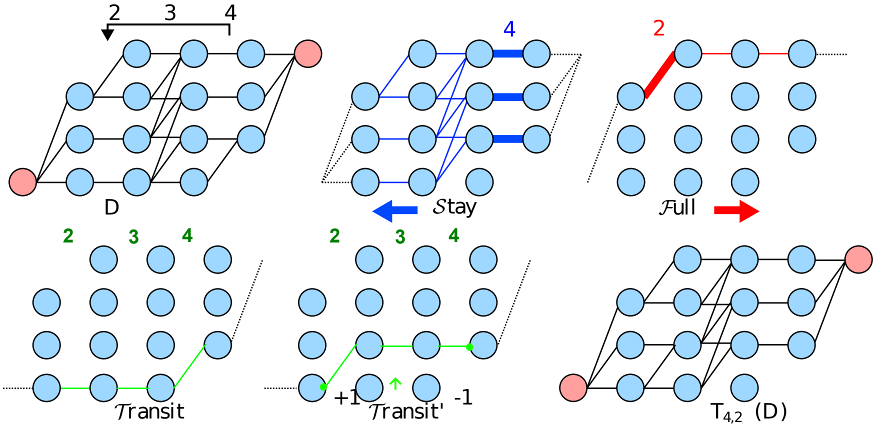
$\mathbf{x}$	$t_{4,2}(\mathbf{x}) = \mathbf{x}$	$t_{4,2}(x) \neq \mathbf{x}$	$t_{4,2}(\mathbf{x})$
	$x_4 = 0$ OR $x_2 = C_2$	$x_4 > 0$ AND $x_2 < C_2$	
01100	•		01100
01110	•		01110
01002	• •		01002
10110		•	11100

Transition $t_{4,2}$ on S

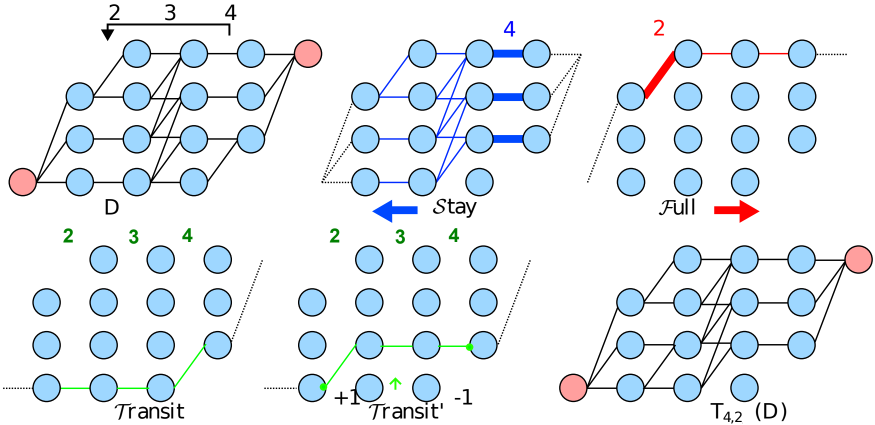
$\mathbf{x}$	$t_{4,2}(\mathbf{x}) = \mathbf{x}$	$t_{4,2}(x) \neq \mathbf{x}$	$t_{4,2}(\mathbf{x})$
	$x_4 = 0$ OR $x_2 = C_2$	$x_4 > 0$ AND $x_2 < C_2$	
01100	•		01100
01110	•		01110
01002	• •		01002
10110		•	11100

- $t_{4,2}(S) = \{(0,1,1,0,0), (0,1,1,1,0), (0,1,0,0,2), (1,1,1,0,0)\}$

Compute $T_{4,2}(D)$ on D



Compute $T_{4,2}(D)$ on D



- Complexity in $O(KM^2)$ compared to $O(M^{K-1})$ (transition on a set of states)

Perfect sampling algorithm

Perfect Sampling with States (PSS)

1. $n \leftarrow 1$
2. $t \leftarrow t_{U_{-1}}$
3. While $|t(\mathcal{S})| \neq 1$
4. $\quad n \leftarrow 2n$
5. $\quad t \leftarrow t_{U_{-1}} \circ \ldots \circ t_{U_{-n}}$
6. Return $t(\mathcal{S})$

Perfect Sampling with Diagram (PSD)

1. $n \leftarrow 1$
2. $T \leftarrow T_{U_{-1}}$
3. While $|\psi(T(\mathcal{D}))| \neq 1$
4. $\quad n \leftarrow 2n$
5. $\quad T \leftarrow T_{U_{-1}} \circ \ldots \circ T_{U_{-n}}$
6. Return $\psi(T(\mathcal{D}))$

Perfect sampling with diagram

Theorem

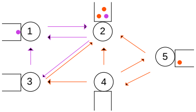
Algorithm PSD terminates in finite expected time and produces an exact sample from the stationary distribution.

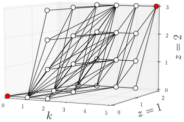
Sketch of proof

- $\psi(\mathcal{D}) = \mathcal{S}$
- Transitions preserve inclusions and coupling
- There exists a finite sequence of transitions $T = T_{i_\rho, j_\rho} \circ \cdots \circ T_{i_1, j_1}$ such that $|\psi(T(\mathcal{D}))| = 1$

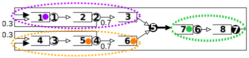
Contributions: perfect sampling of closed queueing networks

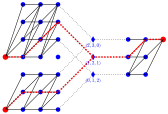
- Multiclass networks [QEST, '15]





- Monoclass with synchronization





- Implementation:
  - Monoclass: `Clones`, Matlab Toolbox [*Best Tool Paper Award* at ValueTools'14]
  - Multiclass: `MClones`, package Python [ROADEF '16]

Motivations

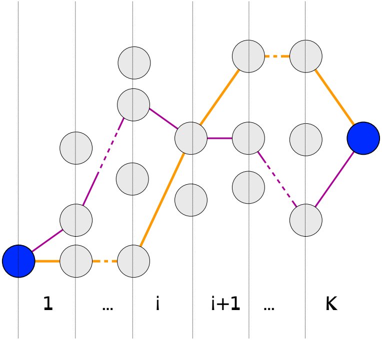
First motivations:

- Generalize the diagrams notations (monoclass and multiclass)
- Define generic algorithms (ψ, $|\psi(D)|$)

- Diagrams can be used in a more general context than closed queueing networks
- Link with dynamic programming

Diagram

$S = \{\mathbf{x}, \mathbf{y}\}$



States represent paths that have the following properties:

- Starting at the same node
- Ending at the same node
- Prefix (and suffix) factorization

Memoryless sequence

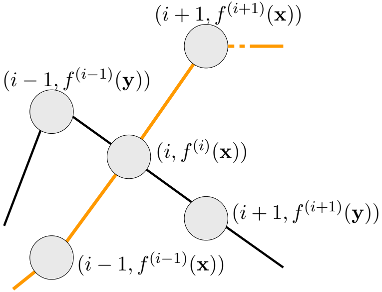
Definition

Let $\mathcal{E}$ (discrete) and $\mathcal{G}$ be two spaces and $\mathcal{E}_k \subseteq \mathcal{E}$.
$(f^{(k)})_{k\in\mathbb{N}}$ is **memoryless** if:

i) function $f^{(0)}$ is constant

ii) $\forall k > 0$,

$$f^{(k)} : \begin{array}{ccc} \mathcal{E}_1 \times \ldots \times \mathcal{E}_k & \to & \mathcal{G} \\ (x_1, \ldots, x_k) & \mapsto & f^{(k-1)}(x_1, \ldots, x_{k-1}) \oplus_k x_k. \end{array}$$



Memoryless space

Definition

Let $(f^{(k)})_{k\in\mathbb{N}}$ a memoryless sequence, the **memoryless space** associated to parameters $K \in \mathbb{N}$ and $M \in \mathcal{G}$:

$$\Omega(K, M) := {f^{(K)}}^{-1}(\{M\}).$$

- $f^{(K)}(\mathbf{x}) = M$ (ending at the same node)
- A diagram encodes a set of state $S \subseteq \Omega(K, M)$

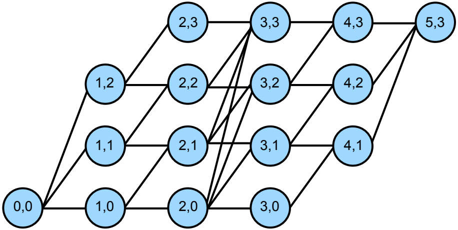
Monoclass closed queueing network

- Memoryless sequence:

$$f_{sum}^{(0)} = 0$$

$$\begin{array}{rcccc} f_{sum}^{(k)} & : & \{0,\ldots,C_1\} \times \ldots \times \{0,\ldots,C_k\} & \to & \mathbb{N} \\ & & (x_1,\ldots,x_{k-1},x_k) & \mapsto & f_{sum}^{(k-1)}(x_1,\ldots,x_{k-1}) + x_k \end{array}$$

- Diagram $K = 5,\ M = 3,\ \mathbf{C} = (2,1,3,1,2)$



Partitions of integer [Application 1]

- Partition of 5: $\mathbf{x} = (1, 0, 0, 1, 0)$, $y = (3, 1, 0, 0, 0)$, ...
- $5 = 1 \times 1 + 1 \times 4 = 3 \times 1 + 1 \times 2 = \dots$

- Diagram representation $O(K^2 \log(K))$

Partitions of integer [Application 1]

- Partition of 5: $\mathbf{x} = (1, 0, 0, 1, 0)$, $y = (3, 1, 0, 0, 0)$, ...
- $5 = 1 \times 1 + 1 \times 4 = 3 \times 1 + 1 \times 2 = \dots$
- Memoryless sequence: $f_{part}^{(0)} = 0$

$$\begin{array}{rccc} f_{part}^{(k)} & : & \mathbb{N}^k & \rightarrow & \mathbb{N} \\ & & (x_1, \dots, x_{k-1}, x_k) & \mapsto & f_{part}^{(k-1)}(x_1, \dots, x_{k-1}) + k * x_k. \end{array}$$

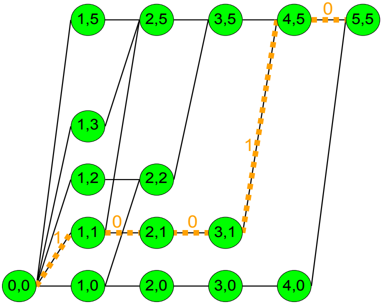
- Diagram representation $O(K^2 \log(K))$

Partitions of integer [Application 1]

- Partition of 5: $\mathbf{x} = (1,0,0,1,0)$, $y = (3,1,0,0,0)$, $\dots$
- $5 = 1 \times 1 + 1 \times 4 = 3 \times 1 + 1 \times 2 = \dots$
- Memoryless sequence: $f_{part}^{(0)} = 0$

$$f_{part}^{(k)} : \begin{array}{ccc} \mathbb{N}^k & \to & \mathbb{N} \\ (x_1, \dots, x_{k-1}, x_k) & \mapsto & f_{part}^{(k-1)}(x_1, \dots, x_{k-1}) + k * x_k. \end{array}$$

- Complete diagram $K = M = 5$



- Diagram representation $O(K^2 \log(K))$

Simulation of a Markov Chain [Application 2]

- Update function (Markov Chain)

$$\begin{cases} X_0 = s_0 \\ X_n = \mathrm{update}(X_{n-1}, U_n) \quad \text{for } 1 \leq n \end{cases}$$

Simulation of a Markov Chain [Application 2]

- Update function (Markov Chain)

$$\begin{cases} X_0 = s_0 \\ X_n = \mathrm{update}(X_{n-1}, U_n) & \text{for } 1 \leq n \end{cases}$$

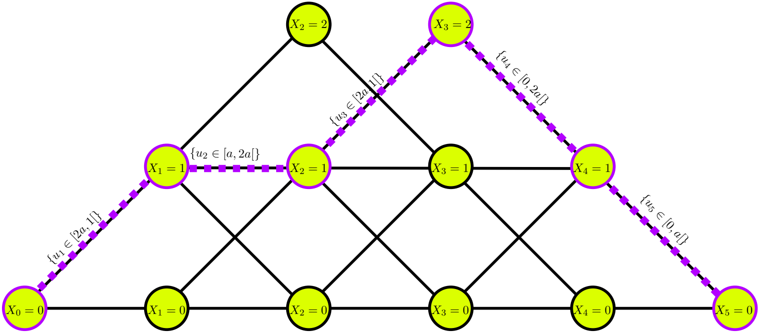
- Sequence without memory

$$f_{mc}^{(k)}(\mathbf{u}) = \begin{cases} s_0 & \text{if } k = 0 \\ \mathrm{update}(f_{mc}^{(k-1)}(\mathbf{u}), u_k) & \text{otherwise} \end{cases}$$

Simulation of a Markov Chain [Application 2]

Example

- $K = 5, M = 0$
- Diagram



Algorithms

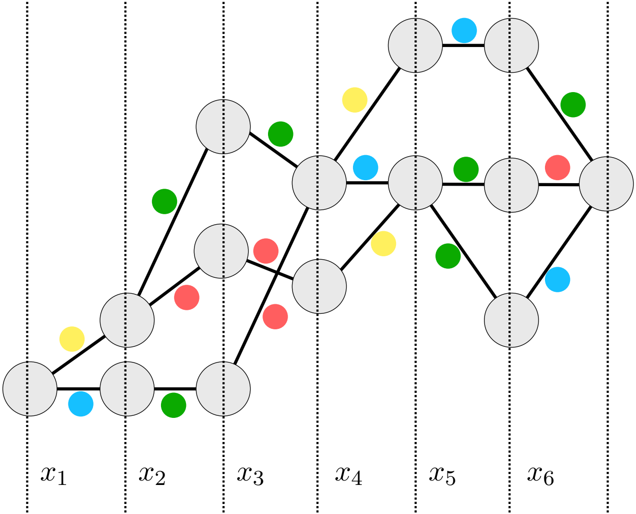
Dynamic programming algorithms:

- *StatesToDiagram* transforms a set of states into a diagram (ϕ)
- *DiagramToStates* transforms a diagram into a set of states (ψ) [Application 2]

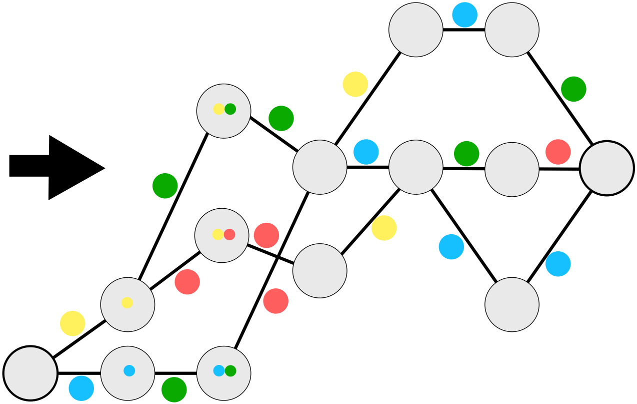
$$
\begin{tikzcd}
S \arrow[d, phantom, sloped, "\subseteq"] \arrow[dr, "\phi"] & \\
\psi(D) \arrow[r, shift left, "\phi"] & D \arrow[l, shift left, "\psi"]
\end{tikzcd}
$$

- *CardStates* computes the number of states represented by a diagram
- *RandState* samples a state according a product form [Application 1]

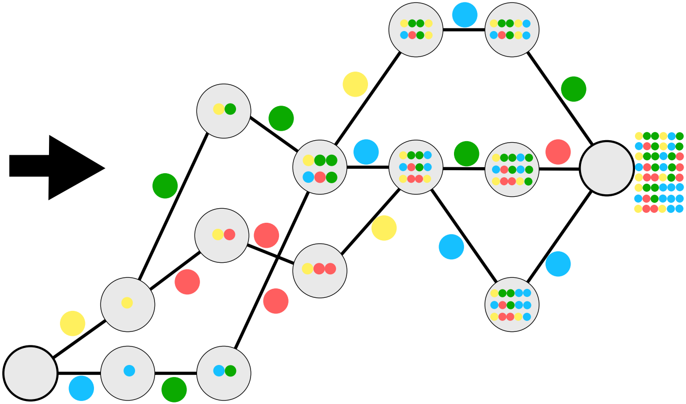
DiagramToStates



DiagramToStates



DiagramToStates



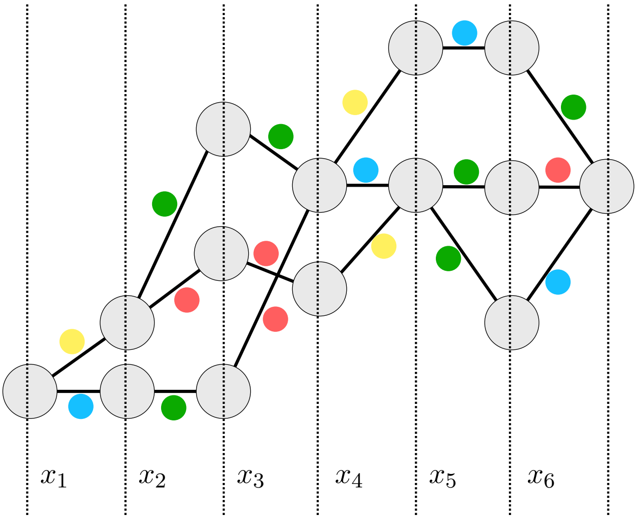
RandState

- Weights $w_k : \mathcal{E}_k \to \mathbb{R}^+$ (given)
- *RandState* produces $\mathbf{x} \in \psi(D)$ according to:

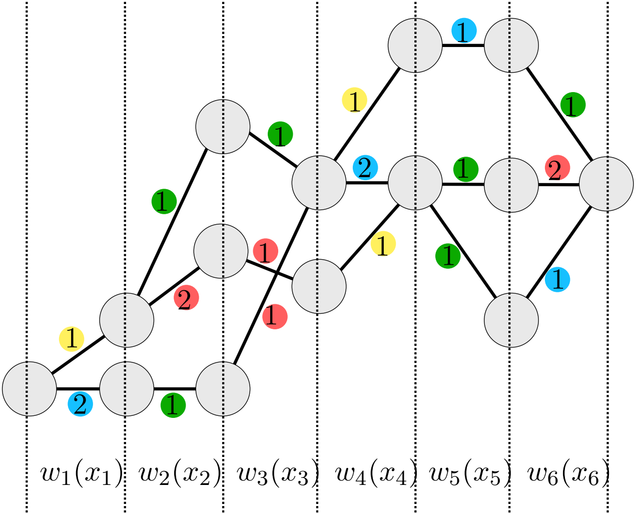
$$p_{\mathbf{x}} = \frac{1}{W} \prod_{k=1}^{K} w_k(x_k) \qquad W = \sum_{\mathbf{x} \in \psi(D)} \prod_{k=1}^{K} w_k(x_k)$$

- 2 steps:
  - Assign weights: arcs and nodes
  - Random walk on the diagram

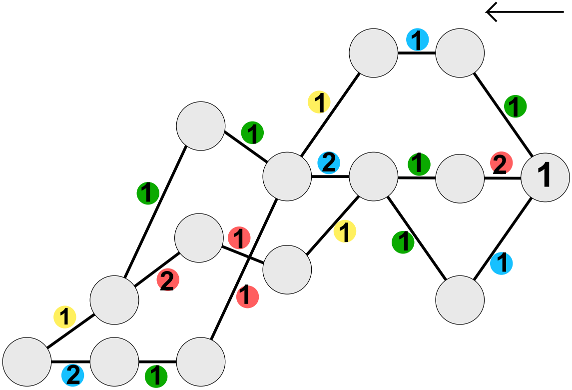
RandState: arcs weight



RandState: arcs weight

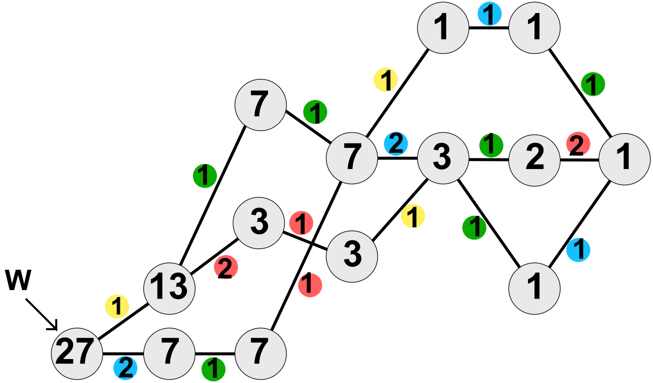


RandState: nodes weight



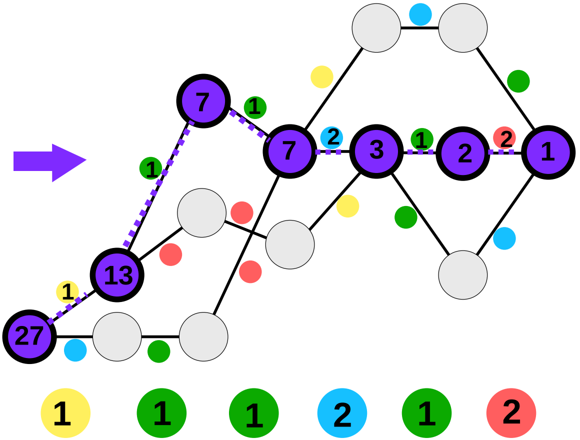
$$w_{\mathcal{N}}(k, m) = \sum_{a=((k,m),(k+1,m'))} w_A(a) w_{\mathcal{N}}(k+1, m')$$

RandState: nodes weight



$$w_{\mathcal{N}}(k,m) = \sum_{a=((k,m),(k+1,m'))} w_A(a) w_{\mathcal{N}}(k+1,m')$$

RandState: random walk



- Probability: $\frac{13}{27}\frac{7}{13}\frac{7}{7}\frac{6}{7}\frac{2}{3}\frac{2}{1} = \frac{4}{27}$

Using RandState

- *RandState* produces $\mathbf{x} \in \psi(D)$ according to:

$$p_{\mathbf{x}} = \frac{1}{W} \prod_{k=1}^{K} w_k(x_k) \qquad W = \sum_{\mathbf{x} \in \psi(D)} \prod_{k=1}^{K} w_k(x_k)$$

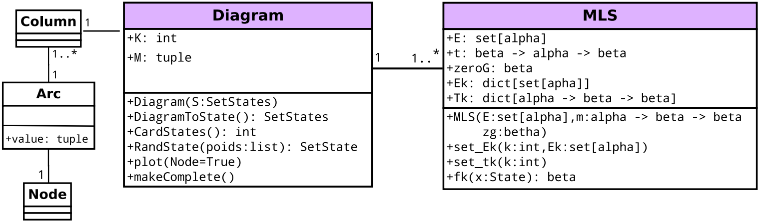
- Complexity: assign weights $O(A)$, random walk $O(K)$
- Sample states for a Gordon-Newell networks
- Uniform generation

$$\forall k, \; w_k(x_k) = 1 \implies p_{\mathbf{x}} = \frac{1}{|\psi(D)|}$$

- [Application 1]

DiagramS

- Package Python, 2 mains classes:



- Using `DiagramS`
  - Define $\mathcal{E}$, $\oplus$ and $0_{\mathcal{G}}$ in `MLS`

$$mls = MLS(range(4), lambda\ x, y : x + y, 0)$$

  - Define parameter K and M in `Diagram`

$$D = Diagram(5, 3, mls)$$

Boltzmann Sampling

- Random generation of combinatorial objects
- Introduced in 2004 by Duchon, Flajolet, Louchard and Schaeffer
- Based on Analytic Combinatorics
- Draws uniformly random objects of size n
  - Without enumerating
  - Size of object is random
  - Statistic control of the size of the objects

Combinatorial class

Combinatorial class $(\mathcal{A}, |.|)$

- Set of objects $\mathcal{A}$
- Size function $|.| : \mathcal{A} \to \mathbb{N}$
- a_n: number of objects of size n, $\forall n \in \mathbb{N}$, $a_n < \infty$

Example

Words of $\{0, 1\}^*$:

- $\mathcal{W} = \{\mathcal{E}, 0, 1, 00, 01, 000, 001, \ldots\}$
- $|.|$ gives the length of the word ($|0010| = 4$)

Generating function

Generating function associated to $(\mathcal{A}, |.|)$:

$$A(z) = \sum_{\alpha \in \mathcal{A}} z^{|\alpha|} = \sum_{n \in \mathbb{N}} a_n z^n$$

Example

Words of $\{0,1\}^*$:

- $\mathcal{W} = \{\mathcal{E}, 0, 1, 00, 01, 000, 001, \ldots\}$
- $|.|$ gives the length of the word ($|0010| = 4$)
- Generating function:

$$W(z) = \sum_{w \in \mathcal{W}} z^{|w|} = \sum_{n \in \mathbb{N}} 2^n z^n = \frac{1}{1-2z}$$

Symbolic method

Construct classes from simpler classes using basics constructions

- **Atomic classes**
  - $(\mathcal{E}, |.|)$: $\mathcal{E} = \{\epsilon\}$, $|\epsilon| = 0$
  - $(\mathcal{Z}, |.|)$: $\mathcal{Z} = \{\zeta\}$, $|\zeta| = 1$
- **Disjoint union**

$$\mathcal{A} = \mathcal{B} \cup \mathcal{C} \implies A(z) = B(z) + C(z)$$

- **Cartesian product**

$$\mathcal{A} = \mathcal{B} \times \mathcal{C} \implies A(z) = B(z)C(z)$$

- ...

Boltzmann sampler

Definition

A **Boltzmann generator** $\Gamma[\mathcal{A}](x)$ of parameter x for $(\mathcal{A}, |.|)$ is a probabilistic algorithm that produces $\alpha \in \mathcal{A}$ w.p. $\mathbb{P}_x(\alpha) = \frac{x^{|\alpha|}}{A(x)}$.

- Uniform generator for objects of same size
- Parameter $x \in]0, R_A[$, control the size

$$\mathbb{P}_x(N = n) = \frac{a_n x^n}{A(x)}$$

Boltzmann sampler - Union

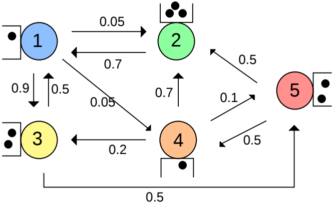
- $\mathcal{C} := \mathcal{A} \cup \mathcal{B}$
- Generating function: $C(z) = A(z) + B(z)$
- $\mathcal{B}er(p)$ return a r.v distributed according a Bernoulli law

$\Gamma[\mathcal{A} \cup \mathcal{B}](x)$

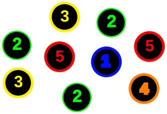
1. $k \leftarrow \mathcal{B}er(\frac{A(x)}{C(x)})$
2. **If** $k == 1$
3. **Return** $\Gamma[\mathcal{A}](x)$
4. **Else**
5. **Return** $\Gamma[\mathcal{B}](x)$

- $\mathbb{P}_{\mathbf{x}}(\gamma) = \frac{x^{|\gamma|}}{C(x)}$

Multiset fixed size



Network K=5, M=9



Multiset size 9

- $\mathcal{B} = \{1, 2, 3, 4, 5\}$, $M = 9$

Multiset fixed size

- Multiset of fixed size: $\mathcal{A} = \mathrm{mset}_M(\mathcal{B})$
- Generating function [Flajolet et al. '07] uses partitions of M

$$A_M(z) = \sum_{\mathbf{p} \in \mathcal{P}_E(M)} A_{\mathbf{p}}(z),\; A_{\mathbf{p}} = \prod_{i=1}^{M} \frac{B(z^i)^{p_i}}{i^{p_i} p_i!},\; \mathbf{p} = (p_1, p_2, \ldots, p_M)$$

Multiset fixed size

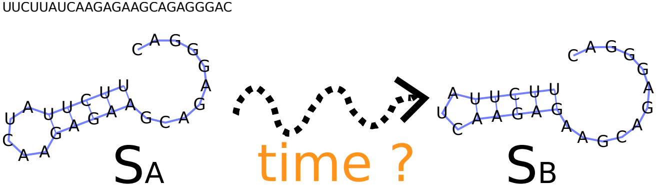
- Multiset of fixed size: $\mathcal{A} = \mathrm{mset}_M(\mathcal{B})$
- Generating function [Flajolet et al. '07] uses partitions of M

$$A_M(z) = \sum_{\mathbf{p} \in \mathcal{P}_E(M)} A_{\mathbf{p}}(z),\ A_{\mathbf{p}} = \prod_{i=1}^{M} \frac{B(z^i)^{p_i}}{i^{p_i} p_i!},\ \mathbf{p} = (p_1, p_2, \ldots, p_M)$$

- *RandState* can be used on a complete diagram of partitions to compute $\Gamma_{\mathrm{mset}_M}[\mathcal{B}](x)$ ($O(M^2 \log(M))$ to assign weight)

Problem: RNA folding kinetic

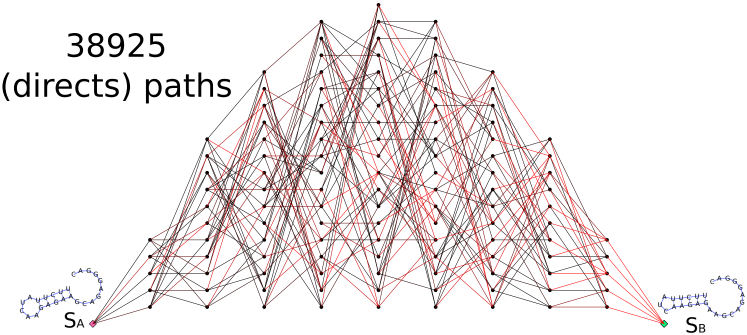
- 2 secondary structures of RNA: s_A and s_B



- GOAL: Estimate the distribution of the hitting time $D_{A \to B}$
- Set of all secondary structure huge

Model and Diagram

- $(Y_n)_{n\in\mathbb{N}}$ an ergodic Markov Chain where $Y_0 = s_A$
- **Paths of size N**: $p = (y_0, y_1, \ldots, y_{N-1}, y_N) \in \Omega^N$ s.t. $y_0 = s_A$ and $y_N = s_B$ (N fixed)

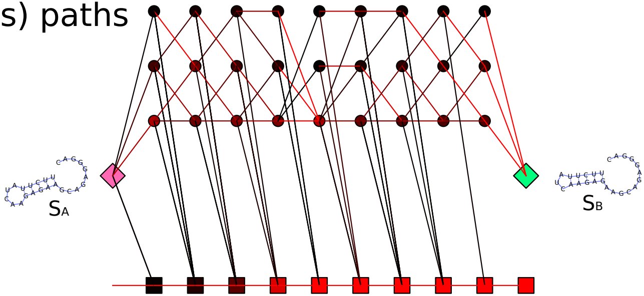


- Goal: estimate $D_{A\rightarrow B}(N)$
- Use DiagramToStates to extract the paths
- Compute the distribution from the set of paths

Nodes and path selection

- Huge number of paths
- Select the "good nodes"
  - Each node as at most *d* successors
  - There are at most *m* nodes in a column

281
(directs) paths



Conclusion

Results (inspired by dynamic programming):

- Perfect sampling for closed queueing network
- Diagram data structure
- 2 examples of applications

Perspectives:

- Perfect sampling using dynamic programming: other networks, other systems ?
- Diagram:
  - Generalized: $\mathcal{E}$ discrete set, continuous ?
  - Parallel computing
  - `DiagramS`
- **RNA folding kinetic**