Courtesy Laurent Chusseau

Marmote& MarmoteMDP
programming Markov Chains & MDPs

Alain Jean-Marie (Inria)
Emmanuel Hyon (U. Paris Nanterre)

PMSMA workshop, Montpellier, 7 June 2023

Contents

1 Motivation and Objectives

Markov modeling

Markov chains on discrete state spaces are useful in many areas of science and engineering:

Operations Research : queueing theory, Markov decision processes, random graphs (*e.g.* PERT), ...

BioInformatics : random sequences, random trees, ...

BioMaths : random population models, epidemic models, ...

Physics : interacting particle models, magnetism, lasers, network science, ...

...

yet...

- scientists outside Stochastic Operations Research do not identify a "Markov" software library that suits their needs;
- even within SOR, Markov modelers continue to do ad-hoc development.

Our Markov chains

Our Markov chains/processes are;

- evolving on a state space $\mathcal{E}$, assumed discrete
- described by a (memoryless) transition rule

$$i \longrightarrow j \qquad \text{with probability } p_{ij} \text{ (discrete time)}$$
$$i \longrightarrow j \qquad \text{with rate } \lambda_{ij} \text{ (continuous time).}$$

Markov modeling consists in

- constructing Markov models:
  - specify state space
  - specify transitions, probabilities/rates
- analyzing them:
  - determine qualitive properties: structure, ergodicity, stability ...
  - compute metrics related with probabilities/distributions, frequencies, times, durations ...

Development

The project currently contains:

- the basis library `MarmoteCore` providing the objects necessary to
- the Markov Chain library per se: `MarmoteMarkovChain`;
- a library dedicated to Markov Decision Processes: `MarmoteMDP`.

Available for three operating systems: linux, MacOS and Windows 10.
Choice of an object-oriented language: `C++` with interfaces with more popular languages.

2
Inside Marmote

Abstractions of Marmote

The programming model is based on just 4 main abstractions (implemented as *classes* in C++)

- Markov chains: `MarkovChain`
- Transitions: `TransitionStructure`
- State Spaces: `MarmoteSet`
- Probability distributions: `Distribution`

and many derived classes.

State spaces in Marmote

A state is always represented as an array of integers.

Marmote provides the MarmoteSet interface with some standard state space implementations:

MarmoteInterval a simple 1-dimensional discrete interval, possibly infinite

MarmoteBox cartesian products of intervals

BinarySequence sequences of bits

Simplex sequences of integers with given total sum

BinarySimplex sequences of bits with given count of ones

State spaces are explored with the help of the state indexing function.

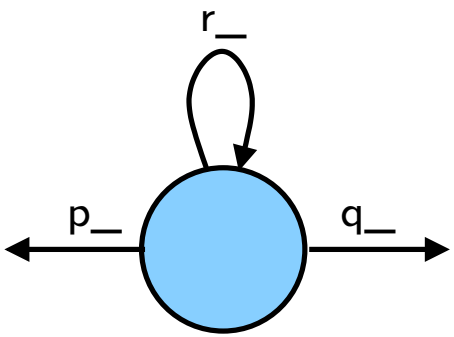
Construction of elaborate state spaces using unions and products.

Transition Structures

- Describes direct transitions between states and their weight.
- Typical implementation will be a (sparse) matrix.
- Possibility of non-matrix implementation.

Example: Implementation of `getEntry()` for the 1-D random walk

```
getEntry(int i, int j)
if ( i == j-1 ) return p_;
else if ( i == j+1 ) return q_;
else if ( i == j ) return r_;
else return 0.0;
```



$\Longrightarrow$ possibility of making simulations over "infinite" state spaces.

3 The MarkovChain object and hierarchy

The Markov Chain object

The `markovChain` object is just a container for its state space and transition structure.

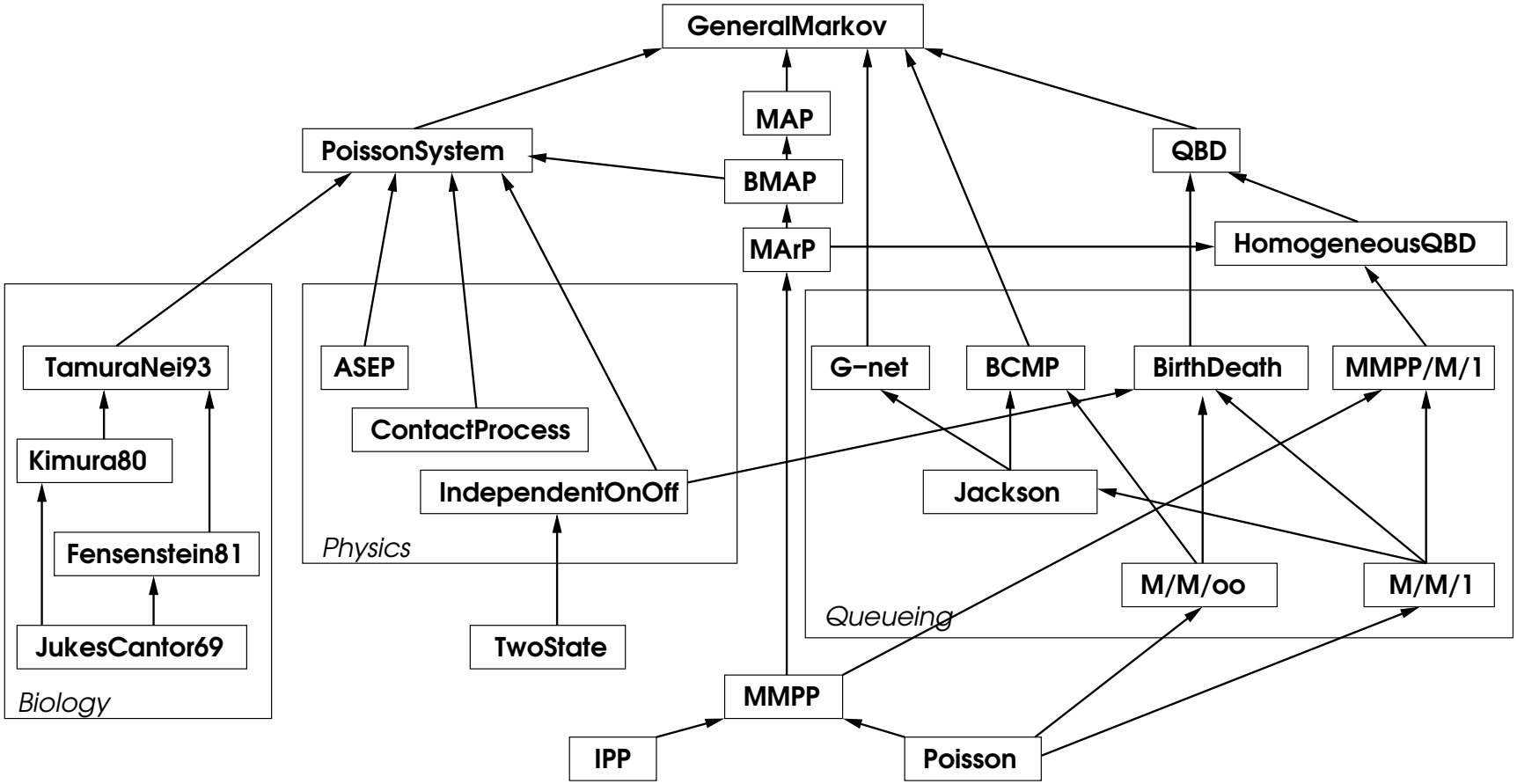
Attributes of `MarmoteMarkovChain`

```
timeType type_;
MarmoteSet* state_space_;
TransitionStructure* generator_;
DiscreteDistribution* init_distribution_;
```

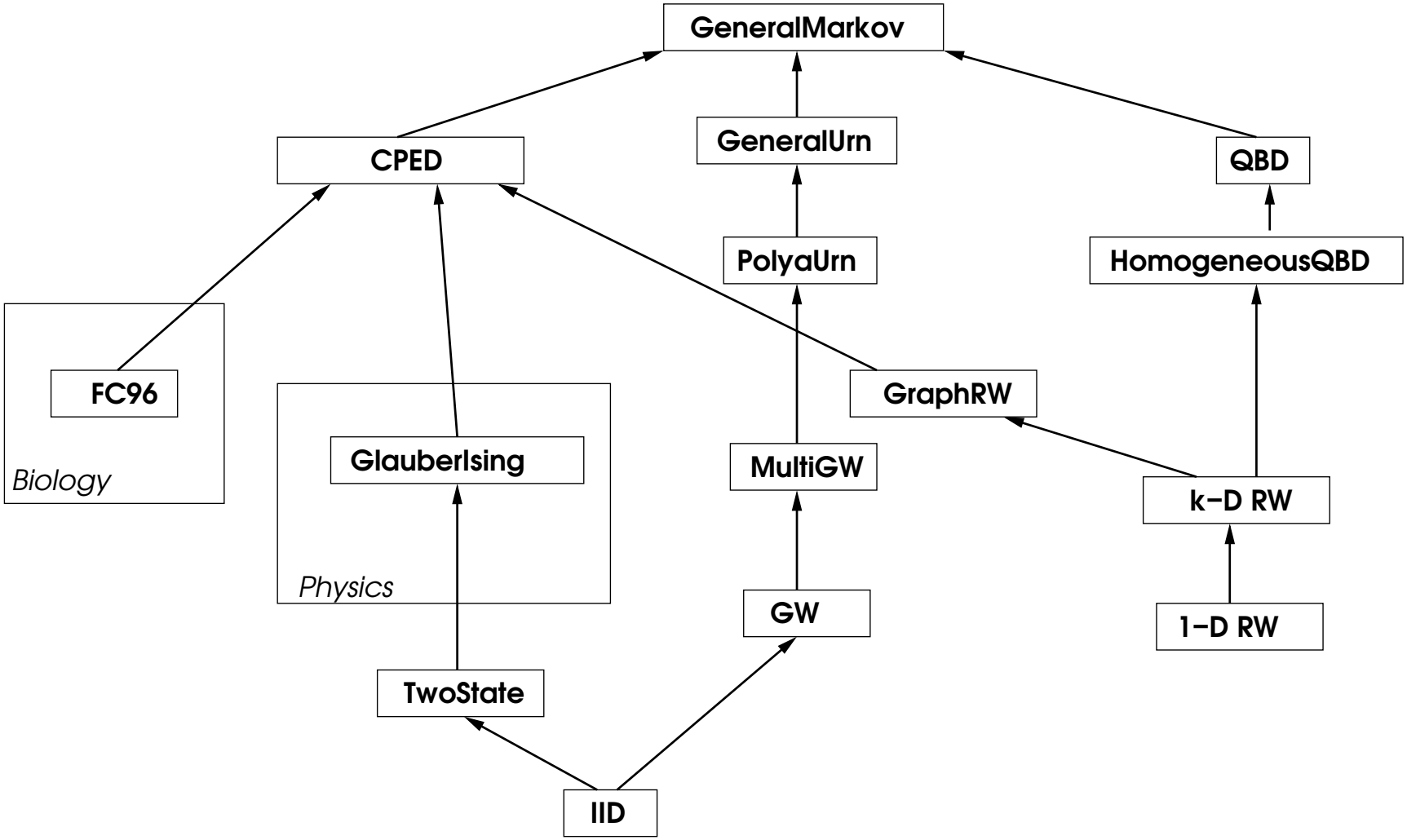
What is more interesting is the possibility to organize families of `MarkovChain` objects in a hierarchy following the inclusion relation. Exploit the principle: The more structure (the fewer parameters), the deeper the analysis

Markov Zoo, continuous time

A hierarchy of Markov models $\Longleftrightarrow$ C++ classes



Markov Zoo, discrete time



Available Solution Methods for `MarkovChain`

- Structural analysis (recurrent/transient classes, period)
- Monte Carlo Simulation (forward)
- Exact sampling from the stationary distribution (backwards)
- Computation of the stationary distribution (various methods)
- Computation of transient distributions
  $\rightarrow$ interface with code written in R
- Hitting times (distribution, average)

4 MarmoteMDP and pyMarmote

The MarmoteMDP library

A library for manipulating Markov Decision Processes.

It allows:

- to solve MDP
  - Computing optimal cost,
  - Computing optimal policy,
- to check structural properties (e.g convexity of Value Function, monotonicity of policy)
- to manipulate Q functions (work in progress)

MDP Objects

Main attributes of a MDP

- State Space: a `MarmoteSet`
- Action Space: a `MarmoteSet`
- Transitions: a vector of `TransitionStructure` objects (e.g a vector of `SparseMatrix`'s
- Reward: a matrix of rewards

There is a MDP object for each criteria:

- `DiscountedMDP`, infinite horizon discounted objective,
- `AverageMDP`, infinite average objective (unichain)
- `TotalRewardMDP`, infinite horizon total reward
- `FiniteHorizonMDP`, finite discounted or total reward

MDP Solving

Implemented algorithms

Currently "from the (Puterman) book" algorithms

- Value Function, Gauss Seidel Value Function,
- Relative Value Iteration,
- Policy Iteration Modified, Gauss Seidel Policy Iteration Modified.

A solution returns an object `Policy`.

Experimental results

- always faster than LP ($\times 60$).
- [Tournaire et al. 2019] $1.6 \cdot 10^6$ states, 3 actions: 38 min
- [Quezada et al 2020] 70728 states, 20 actions, Full Matrices, Finite Horizon 10: building 37 min and solving 28 min.

pyMarmote

Associated with a previous version of `MarmoteCore` (aka `marmoteBase`), there exists a python extension:

`pyMarmoteMDP`

Only on linux.

Available (with numerous examples) at:
`https://webia.lip6.fr/~hyon/Marmote/telechargements.php`

User code is written in python but behind runs the C++ library. Advantage fast but easy to handle.

There is (still) not a (pip) package (but students have used it without difficulties).

5 Resources

Resources

On line resources

Links to access:

- the C++ libraries for `Marmote` and `MarmoteMDP`
  https://marmote.gitlabpages.inria.fr/marmote
- the `pyMarmote` interface of `MarmoteMDP` in python:
  https://webia.lip6.fr/~hyon/Marmote/home.php
  N.B. Uses an old API of `Marmote`.
- the C++ development project:
  https://gitlab.inria.fr/marmote
  N.B. Currently restricted to contributors: to become one, inquire at `Alain.Jean-Marie@inria.fr`.

6 Conclusion

As a conclusion

`Marmote` and `MarmoteMDP` provide an operational environment for

- experimenting with Markov chains, MDP, etc.
- implementing and benchmarking new algorithms.

It is an open development,

- in need of users/testers
- in need of contributors
  - thematic libraries: Semi-Markov, Hidden-Markov, Markov-Game, ...
  - domain-specific: BioInfo, Populations, Reliability/Performability, Data & Markov, ...