T-61.6030 Dynamical models for prediction and decision making: project description

*** *** *** *** *** *** *** *** *** *** *** *** ***

Running large-scale Markov system implementations

The task is to write a few versions of programs to run a Markov chain. Write a program for running a Markov chain with a programming language
(or environment) of your choice. Good candidates are for instance C, C++, Python, Java, Matlab, R or other. You can try different/many languages on each of the tasks. Write the program in a way that allows you to experiment with Markov chains with varying number of states.

You can use random probabilities in the implementations. We are interested in the running times and the feasibility of large-scale Markov chains, rather than on the details of the results.

All the program versions should be timed. In timing, understand the difference between the "wall clock time" and execution time. Learn how to do these in your languages of choice.

Tasks to be completed:

1. In the first version, implement a Markov chain without relying on any libraries or ready made solutions. Just use the language constructs in the language you have chosen.

Report: running times as a function of the size of the system (#states).

2. Modify the previous version and make use of a library or a package and see if you can make the solution any faster. Possible solutions include BLAS, GNU Scientific library (GSL), Matlab packages etc.

Report: running times as a function of the size of the system (#states), and used library.

3. Assume sparsity of the transitions, that is, imagine a Markov chain where a state describes a road segment in the traffic network. From a road (once you are at the crossing), you can reach only a limited number of other roads, this number being typically 2-4. Implement a Markov chain taking into account this type of a constraint.

Report: running times as a function of the size of the system (#states) and degree of assumed sparsity.

4. Do the above, but by using a library or some help construct (sparse matrix library) in the solution.

Report: running times as a function of the size of the system (#states) and degree of sparsity.

5. As an extra task, study simple, high-level ways to parallelize the code. As candidates, look at OpenMP or MPI. See if you can improve the
running times.

Report: running times as a function of the size of the system (#states), and used library, setup on the resources (#processors/threads used). Also report your familiarity with the system and the time you invested.

You should report the following for each task:

Program code in the language of your choice and a short description how to run the code.

Vary the size of the Markov chain from very small (a few) to very large (thousands or hundred of thousands). Run the system for 1000 steps, that is, repeat the matrix-vector product with the matrix being the transition matrix and the vector being the probability for the state. Run systems of increasing sizes until they fail, report the mode for failure (out of memory, I was bored to wait for results).

In the end, we'll collect all the results together and see how the

programming environments/languages compare with each other, and how
much there is variability between the implementations. Lessons learned could be written up as a technical contribution.

*** *** *** *** *** *** *** *** *** *** *** ***