ScenTrees.jl: A Julia Library for Generating Scenario Trees and Scenario Lattices for Multistage Stochastic Programming

Kipngeno Kirui* Alois Pichler* Georg Ch. Pflug†

November 5, 2019

Summary

In multistage stochastic optimization we are interested in approximations of stochastic processes. In this setting, stochastic processes have random and uncertain outcomes and therefore decisions must be made at different stages of the process. It is generally intractable to solve mathematical programs with uncertain parameters described by an underlying distribution. The common approach is to form an approximation of the original stochastic process or underlying distribution by discretization. The procedure of discretizing a stochastic process is called *scenario tree generation*. We depict the possible sequences of data for this processes in form of a *scenario tree* in the case of a discrete time stochastic process and a *scenario lattice* for Markov processes.

Since the paper of Høyland and Wallace (2001), scenario tree generation has been used to solve various multistage stochastic problems in the industry and academia. Various authors including Pflug (2001), Kovacevic and Pichler (2015) and Pflug and Pichler (2016) have come up to add and improve various ideas into the process of generating scenario trees. However, there is no fast and open-source implementation of the algorithm that has been available in the public domain for various users to appreciate. Instead, various researchers and industries have been forced to code their own implementations in a variety of languages they like themselves. This has limited many researchers and industries who would not like to code themselves from generating scenario trees with available implementations. Many researchers and industries also would not get the taste of comparing ideas against their own implementations and hence they may end up trusting their own results and implementations and maybe there is a better implementation.

The natural question when dealing with scenario generation for multistage stochastic optimization problems is how to approximate a continuous distribution of the stochastic process in such a way that the distance between the initial distribution and its approximation is minimized. We start with an initial scenario tree and use stochastic approximation to improve the scenario

*University of Technology, Chemnitz, Germany
†University of Vienna

tree. To quantify the quality of the approximation, we use the process distance between the original process and the final scenario tree (Pflug (2009)). Process distance extends and generalizes the Wasserstein distance to stochastic processes. It was analyzed by Pflug and Pichler (2012) and used by Kovacevic and Pichler (2015) directly to generate scenario trees.

Generally, we consider a stochastic process X over a discrete time, i.e., $X = (X_0, X_1, \ldots, X_T)$ where X_0 is deterministic. A scenario tree is a discrete time and discrete state process approximating the process X. We represent a scenario tree by $\tilde{X} = (\tilde{X}_0, \tilde{X}_1, \ldots, \tilde{X}_T)$.

A scenario tree is a set of nodes and branches used in models of decision making under uncertainty. Every node in a scenario tree represents a possible state of the stochastic process at a particular point in time and a position where a decision can be made. Each scenario tree node has only a single predecessor, but can have multiple successors. This makes the difference with scenario lattices since nodes of scenario lattices can have multiple predecessors. A scenario tree is organized in levels which corresponds to stages $0, 1, \ldots, T$. Each node in a stage has a specified number of predecessors as defined by the branching structure of the scenario tree. An edge from a node indicates a possible transition of the uncertain variables from that state. The first node of a scenario tree is called the root node and the set of nodes in the last stage are called the leaves. Any possible path from the root node to any of the leaf nodes is called a *trajectory* (also called a *path* or a *scenario*).

ScenTrees.jl is a Julia (Julia (2019)) library for generating scenario trees and scenario lattices which can be used, for example, for multistage stochastic optimization problems. The theory and design of the *ScenTrees.jl* library follows Algorithm 5 in Pflug and Pichler (2015). The library's design allows us to obtain a fast code with high flexibility and excellent computational efficiency. The design choices were highly motivated by the properties of the Julia (Bezanson et al. (2017)) language. It starts with a tree which is a qualified guess by the user or an expert opinion and iterates over the nodes of the tree updating them with scenarios drawn from a user-defined distribution. In this way, the approximating quality of the tree is improved for each scenario generated. The iteration stops when the predefined number of scenarios have been performed.

Main features of the library

The stochastic approximation framework allows ScenTrees.jl to be generally applicable to any stochastic process to be approximated. The following are key features that ScenTrees.jl provides. Implementation details and examples of usage can be found in the software's documentation.[1]

(i) *Generation of scenario trees and scenario lattices from stochastic processes using the stochastic approximation algorithm*: Here, the structure of the scenario tree or the scenario lattice is fixed in terms of the branching vector then stochastic approximation is used to improve the states of the nodes considering all the data available for every approximation. This improvement goes on until a pre-specified number of iterations have been performed and then the process distance is calculated.[2]

[1]Documentation: https://kirui93.github.io/ScenTrees.jl/latest
[2]Tutorial4: https://kirui93.github.io/ScenTrees.jl/latest/tutorial/tutorial4/

(ii) *Generation of scenarios based on data*: This is a non-parametric technique for generating new samples from a given data. Here we have data from some observed trajectories of a scenario process with an unknown distribution and we want to use conditional density estimation to generate new but different samples based on the above trajectories. The new samples can then be used to generate scenario trees or scenario lattices using the stochastic approximation procedure.[3]

Example: Scenario generation from observed trajectories

We want to use the concept of density estimation and stochastic approximation to generate scenario trees and scenario lattices for a (1000×5) dimensional data. The first step is to load the library and the data into Julia as follows:

```
# Load the package
julia> using ScenTrees
# Load the csv data from a directory
julia> df = CSV.read("../RData.csv");
# Convert the DataFrame into a Matrix
julia> data = Matrix(df);
```

The following shows an example of a non-Markovian trajectory generated from the data using conditional density estimation:

```
julia> Example = KernelScenarios(data,Logistic;Markovian=false)()
[1.5595,0.8150,1.5058,2.6475,4.6137]
```

The generated data has a length equal to the number of columns of the original data. These generated trajectories are the ones we use to approximate a scenario tree and a scenario lattice in the following subsections.

Approximating with a scenario tree

Consider a scenario tree with a branching vector $[1, 3, 3, 3, 2]$ (cf. Fig.1). We approximate this process using $1,000,000$ number of iterations as follows:

```
julia> ApproxTree = TreeApproximation!(Tree([1,3,3,3,2],1),
KernelScenarios(data,Logistic;Markovian=false),1000000,2,2);
julia> treeplot(ApproxTree)
julia> savefig("ApproxTree.pdf")
```

[3]Tutorial41: https://kirui93.github.io/ScenTrees.jl/latest/tutorial/tutorial41/

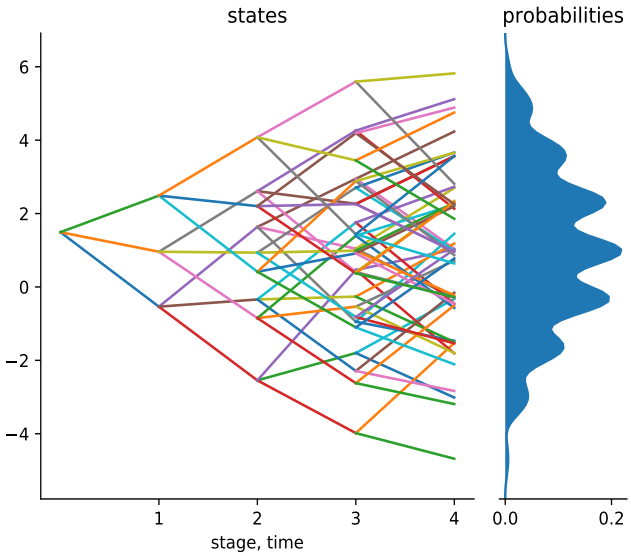

Figure 1: Scenario tree with branching structure $(1, 3, 3, 3, 2)$ `fig1`

The number of possible trajectories in the scenario tree equals its number of leaves. In Figure 1, there are 54 possible trajectories as the total number of leaves are $(1 \times 3 \times 3 \times 3 \times 2) = 54$. One important thing to note also is that a tree with more branches will have a better approximation quality. The algorithm returns the multistage distance between Figure 1 and the original stochastic process as $d = 0.23092$.

Approximating with a scenario lattice

Consider a scenario lattice with a branching vector $[1, 3, 4, 5, 6]$ (cf. Fig. 2). Clearly, this scenario lattice has 5 stages as shown by the number of elements in the branching vector, which is equal to number of columns in the data. We consider $1,000,000$ iterations for the stochastic approximation algorithm.

```
julia> ApproxLattice = LatticeApproximation([1,3,4,5,6],
KernelScenarios(data,Logistic;Markovian=true),1000000);
julia> PlotLattice(ApproxLattice)
julia> savefig("ApproxLattice.pdf")
```

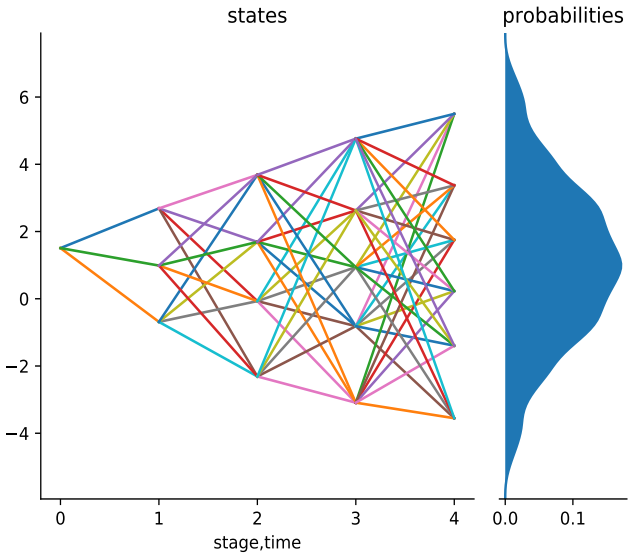

Figure 2: Scenario lattice with branching structure $(1, 3, 4, 5, 6)$ fig2

The number of nodes in the scenario lattice is equal to the sum of elements in the branching structure, i.e., $1 + 3 + 4 + 5 + 6 = 19$ nodes. The scenario tree in Fig. 1 has 94 nodes and 54 possible scenarios while the scenario lattice in Fig. 2 has 19 nodes and $(1 \times 2 \times 3 \times 4 \times 5) = 360$ scenarios. This shows that a scenario lattice with fewer number of nodes can have many possible trajectories than a scenario tree with more number of nodes. The algorithm returns the multistage distance between the scenario lattice and the original process as $d = 1.1718$.

Acknowledgments

This work was supported by the German Academic Exchange Service (DAAD).

References

julia J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017. doi:10.1137/141000671. 2

Hoyland2001 K. Høyland and S. W. Wallace. Generating scenario trees for multistage decision problems. *Management Sceince*, 47:295–307, 2001. doi:10.1287/mnsc.47.4.295.9834. 1

JULIA2019 Julia. *Julia version 1.2*. Julia Computing, MIT, 2019. URL https://docs.julialang.org/en/v1. 2

ovacevicPichler R. M. Kovacevic and A. Pichler. Tree approximation for discrete time stochastic pro-

cesses: a process distance approach. *Annals of Operations Research*, pages 1–27, 2015. doi:10.1007/s10479-015-1994-2. 1, 2

Pflug2001 G. Ch. Pflug. Scenario tree generation for multiperiod financial optimization by optimal discretization. *Mathematical Programming*, 89:251–271, 2001. doi:10.1007/s101070000202. 1

Pflug2009 G. Ch. Pflug. Version-independence and nested distributions in multistage stochastic optimization. *SIAM Journal on Optimization*, 20:1406–1420, 2009. doi:10.1137/080718401. 2

flugPichler2011 G. Ch. Pflug and A. Pichler. A distance for multistage stochastic optimization models. *SIAM Journal on Optimization*, 22(1):1–23, 2012. doi:10.1137/110825054. 2

rDynScenarioGen G. Ch. Pflug and A. Pichler. Dynamic generation of scenario trees. *Computational Optimization and Applications*, 62(3):641–668, 2015. doi:10.1007/s10589-015-9758-0. 2

flugPichler2016 G. Ch. Pflug and A. Pichler. From empirical observations to tree models for stochastic optimization: Convergence properties. *SIAM Journal on Optimization*, 26(3):1715–1740, 2016. doi:10.1137/15M1043376. 1