Bayesian Network Adaptor

Summary

The **Bayesian Network adaptor (BN Adaptor)** sets up a Bayesian network based decision support system (DSS) for coastal hazards and impacts in hotspot areas, according to the framework developed in task 3.3 of the RISC-Kit project (www.risckit.eu).

The BN Adaptor is an executable that uses

1. netCDF files of storm simulations
2. text files indicating receptor locations
3. JSON files specifying the structure of the BN
4. a JSON file specifying the vulnerability relationships for the hotspot area

to create

1. a GeNIe file (http://www.bayesfusion.com/) containing the Bayesian network based DSS

The BN Adaptor can be used both in FEWS and outside FEWS.

System Requirements

The BN Adaptor has been developed in c++.

- For Windows the BN Adaptor consists of an executable and a number of dll's. Moreover, it depends on Microsoft Visual Studio 2010 express (http://microsoft-visual-cpp-express.soft32.com/).
- For Linux the users have to compile the source code themselves to create an executable. It depends on the ***netcdf***, ***zlib*** and ***boost*** libraries. Run the provided ***compile.sh*** script in the source directory.

Configuration

This section contains the details on configuration in FEWS or outside FEWS.

Directory Structure

The BN Adaptor works with relative paths to read and write data. Initially, users have to create (manually) a directory Risckit/Modules/Genie/<case_study_name>/ which contains:

- input/
- model/
  - BN Adaptor.exe & related .dll's

- output/
    - webviewer/
- trainingData/
- workDir/
    - info.json
    - BCnodes.json
    - Rnodes.json
    - Hnodes.json
    - Cnodes.json
    - Mnodes.json
    - <receptor>.txt
    - vulnerabilities.json
    - shapefiles/
        - areas.shp & related GIS files

In principle *input/* contains the latest simulation results of the hydro/morphodynamic model which transforms offshore boundary conditions to onshore hazards (e.g. XBeach; in the remainder the term "hazard transformation model" is used.) and *trainingData/* contains all previously run simulations, which should eventually amount to the "100 simulations" of task 5.3.

If the user works with the FEWS system the folders *input* and *trainingData* remain empty, as they are filled by the general adaptor (GA). The corresponding actions that the GA needs to take are described in the next section. If the user works outside the FEWS system, the folder *input* is obsolete and *trainingData* need to be filled manually, which is described in the section thereafter.

Working with FEWS

The CSO runs the "100" simulations within FEWS and the Bayesian network is the last model of the model train that is operated by the GA (the site specific config file) following the hazard transformation model. To use the BN Adaptor in order to create the Bayesian network based DSS, the user has to add three actions to his/her site specific GA. In general the model output of the hazard transformation model is imported to FEWS after each run. First, the output is transformed to two separate files which are saved in *input/: hazardbc.nc* (or *hazardbc_<measure>_<bin>.nc* if a structural measure is in place) contains the boundary conditions and *hazard.nc* contains 'gridded' output of the onshore hazards. These netCDF files must follow strict format rules in order to comply with the requirements of the BN Adaptor (see section netCDF files). Second, a batch file copies the two files to a new directory with an arbitrary but unique name in *trainingData/*. Third, the GA calls the BN Adaptor, which creates *output/BN<hotspotname>.dsl*.

Example configurations can be found at the bottom of this page (General Adaptor Run Example Configuration).

Working outside FEWS

Working outside FEWS is conceptually the same as working inside FEWS. The difference is that the tasks of the GA will be executed manually. The user will run the "100 simulations" of task 5.3. with the hazard transformation model only and collect all model outputs. Matlab scripts similar to hazardbc_example.m and hazard_example.m (created by Haris) can be used to transform those outputs to *hazardbc.nc* and *hazard.nc* files of the right format and to save them in separate directories in *trainingData/*. Then the BN Adaptor.exe can be run and will create *output/BN<hotspotname>.dsl*.

Input and Output File Formats

This section describes the formats of files that need to be placed in *model/* and *workDir/* by the user before the BN Adaptor is run and that are written to *input/* and *trainingData/* by the GA and its batch file. Note that the user develops his/her own GA and has to ensure that it writes files of the correct format to *input/* and *trainingData/*. It also describes the files that are written by the BN Adaptor to *output/*.

JSON files

The JSON files are the backbone of the Bayesian network. The BN Adaptor reads JSON files from *workDir/* (and in the future: writes output to *output/*) and they constitute the link between the simulation data and the BN Adaptor. They

- contain all variable definitions (e.g. name, type, discretisation)
- define the structure/design of the Bayesian network (i.e. causal relationships between the variables)
- provide the BN Adaptor with information on the simulations (e.g. grid type and size)

The required JSON files have to be created manually by the user. They are *info.json*, *BCnodes.json*, *Rnodes.json*, *Hnodes.json*, *Cnodes.json*, *Mnodes.json* and *vulnerabilities.json*. Their format is explained below and examples are provided.

JSON files can be created with any text editor. They are built on two structures: (1) a collection of name/value pairs and (2) an ordered list of values. Details can be found at http://json.org/.

Info file

The file info.json contains six or seven fields, depending on the grid being unstructured or structured.

- "name" specifies the filename of the resulting BN
- "grid_type" is either "structured" or "unstructured"
- "hazardbc_time" denotes the time dimension of the netCDFs and should be 1 (not desired, but possible: hazardbc_time is equal to the constant number of time steps in hazardbc.nc)
- "hazardbc_stations" should be 1 and denotes the location of the boundary condition
- "hazard_time" should be 1 and denote the time dimension of the netCDFs

- In case of an unstructured grid: "hazard_stations" is used to specify the number of model grid points
- In case of a structured grid: "hazard_row" and "hazard_col" are used to specify the dimensions of the model grid

Structure files

The structure files contain the variable definitions (including the links between them) for the Bayesian network. Variables of the same category are defined in the same file. Because there are five categories, i.e. boundary condition (BC), receptor (R), hazard (H), impact (C) and measure (M), there are five structure files. Each variable is defined by a name/value pair. The name is a dummy name, consisting of the category and an index, e.g. BC1 (boundary condition 1) or H1_R1 (hazard 1 for receptor 1). The paired values are JSON objects themselves with at least five name/value pairs on their own:

  - "ID" contains the actual variable name. Consistent naming of the variables across all files (netCDF, JSON, text) is crucial:
    - For BC variables the ID must correspond to the FEWS ID of a variable in the hazardbc.nc file.
    - For R variables the ID points to the name of a text file.
    - For H variables the name has two parts which are divided by a single underscore. The first part indicates the type of hazard and must correspond to the name of that variable in the hazard.nc files. The second part indicates for which receptor type the hazard is calculated and hence it must correspond to the ID of a receptor specified in Rnodes.json.
    - For I variables the name again has two parts divided by a single underscore indicating the type of impact and the receptor affected. Both parts also reappear in the vulnerabilites.json.
    - For M variables there are no special requirements as yet.
  - "title" is what is displayed in Genie. It may contain spaces.
    - Not allowed characters: ( ), €, £
    - Allowed characters: [ ], %
  - "bins" is an array containing the bin values itself or the interval boundaries for each bin, depending on the value in "nobins". The values can either be numeric or a string starting with a letter without any spaces. If they are numeric, they must be strictly increasing.
  - "nobins" is an integer defining the number of bins (or states) in the BN. If "nobins" equals the length of the "bins" array, each value in "bins" is a bin. If "nobins" is one minus the length of the "bins" array, the values in "bins" are taken to be the interval boundaries of each bin.
  - "parents" is an array of the IDs of parent variables or "None". Note that even a single parent has to be in array format, i.e. written within [ ])

Some categories have additional (optional) keywords. This is illustrated in the following examples, which yield this BN structure: BNdemo.dsl

- BCnodes.json for the boundary conditions
- Rnodes.json for the receptors
  - "type": "latent" is an optional keyword for latent receptors. Since latent receptors are not displayed, the field "title" is not needed.
- Hnodes.json for the hazards
  - "aggregation" describes the method by which data will be extracted from netCDF files. This is only relevant for hazard variables. It can have either "mean", "max" or "min" (note e.g. that maximum erosion is min sedero!) as a value, indicating whether the average or maximum value of surrounding hydraulic grid points should be computed. Default is "mean".
- Cnodes.json for the impacts (consequences)
- Mnodes.json for the measures
  - "effectiveness" is an additional key that can be set to take into account that selected measures are not always (effectively) adopted by the population. This will introduce an additional node "Implementation" which has the two states "effective and "not effective" indicating the percentages of effective and not effective implementation of the measure. The value of effectiveness can be determined with the methods outlined in this report. Link to report: CSO Guideline for integrating DRR measures in BN. (Note: if effectiveness is set, then "bins" must have the values "No" and "Yes")
  - For more implementation details of measures see Implementing Measures as well as the next section on the vulnerability file and Instructions: Creating a vulnerability mapping.

Vulnerability file

Similar to the other JSON files, the vulnerabilities.json defines a number of vulnerabilities by a a name/value pair, where the name is <impactID_receptorID> and the value is a JSON object with two keys:

- "type" should be "categorical" (extensions are planned)
- "mapping" contains a 1D integer array that maps all possible combinations of parent nodes' states to the state of the impact variable

An example corresponding to the demoBN is given here vulnerabilities.json and detailed explanations on creating the mapping are given in the Instructions: Creating a vulnerability mapping.

netCDF files

The netCDF files contain data from the hazard transformation model. They do not contain time series data, but a single summary statistic of the time series, e.g. maximum, mean, minimum, or mode. The hazard boundary conditions (e.g. maximum wave height, predominant wave angle, etc for a single location at the 20m water line) are stored in a file with name "hazardbc.nc", while the onshore hazards (e.g. gridded data of maximum flood depths, maximum flow velocities, maximum erosion, etc.) are stored in a file with name "hazard.nc". The format of the two files is a different, however for both applies that

- variable names must be consistent with the "ID" in the JSON files
- variable names must not contain underscores

hazardbc.nc file

The variables in these files should have the dimensions time=1 and stations=1. The dimensions must be cross-referenced with the info.json file and the variable names must be identical to the "IDs" in BCnodes.json. The format is the same for structured and unstructured grids.

Currently the BN Adaptor can cope with time dimensions larger than 1, provided that all but one time step contain fill values and not real data, and that all hazardbc.nc have the same time dimension. In this case the "hazardbc_time" in info.json needs to correspond to the actual time dimension of the netCDF. Nevertheless, this format may lead to errors and it is highly recommended to work with a single time step.

hazard.nc file

The variables in these files should have the dimensions time=1 and, in case of an unstructured grid, stations=n or, in case of a structured grid, row=m and col=l, where n,m,l specify the dimensions of the model grid of the hazard transformation model. Again, the dimensions must be cross-referenced with the info.json file and the variable names must be identical to the first part of the "IDs" in Hnodes.json.

Here are example files for a structured grid and for an unstructured grid.

Text files

The text files provide information on the receptors' spatial distribution across the case study site. For each <receptor> variable defined in Rnodes.json there must be a corresponding <receptor>.txt file which relates all individual receptors to their sub-areas and grid points of of the hydraulic model (e.g. XBeach). The files should be tab-separated and contain, in this order (!), the columns areaID, receptorID and gridpointID. Moreover, the data must be sorted such that the receptorIDs are ascending.

Example from the demoBN are ResBuildings.txt and People.txt. (To create People.txt it has been assumed that there are two people in each building and that they can be found at this location at any time.) Note that there are duplicates of receptorID, because individual receptors may be affected by multiple surrounding grid points. For a step by step explanation of creating this file, see the page Tutorial: Creating a <receptor>.txt file. The approach can be slightly modified to obtain a txt file for a "latent receptor", i.e. a buffer zone around a receptor, which is computationally treated as a receptor, but not explicitly represented as such in the BN. See Tutorial: Creating a <receptor>.txt for a latent receptor

Shape files

The shape files are not required for the BN Adaptor, but the one defining the areas, e.g. "areas.shp", is relevant for the webviewer. In principle this shape file is identical to the one that has been used to create the <receptor>.txt files. For the webviewer this shape file

- must have arealD's which are integers 1, ..., n where n is the total number of areas in the case study
- must be exported to the geoJSON file format (Check validity at http://geojsonlint.com/ !)

The resulting geoJSON file defining the areas as well as all JSON files that are created in the output/webviewer directory are input for the webviewer.

Testing Input Files & Trouble Shooting

Before running the the BN Adaptor the correct format of all input files should be verified. Deviations from the prescribed format will cause the BN Adaptor to crash, with or without useful error message. To aid with verification

- http://jsonlint.com/ can be used to verify that the JSON files are indeed valid json files with proper syntax, e.g. no missing commas.
- code_for_receptor_txt_files_v2.m should be used to ensure the right file format for <receptor>.txt files and to check that the gridIDs passed on to the BN Adaptor indeed correspond to the right locations
- BN_input_check.py should be used to check the format of the JSON files

Downloads

Windows

*** NEW ***

- Version 1.2 (complete)

*** OLD ***

- Version 1.1
- Version 1.0 (complete)
- Version 1.0 (Structure only, no training)
- Version 1.0 (Training limited to BC nodes)
- Version 1.0 (Training limited to BC and M nodes)
- Version 1.0 (Training limited to BC, M and R nodes)
- Version 1.0 (Training limited to BC, M, R and H nodes)

Linux

*** NEW ***

- Version 1.2 (only source code)

*** OLD ***

- Version 1.1 (only source code)
- Version 1.0 (only source code)

Examples

Bayesian DSS Examples

This section contains three examples of the in- and output files of the BN Adaptor. The zip files contain the entire directory structure described above as well as the BN Adaptordaptor.exe. If the executable is run, the GeNIe file (.dsl) in output/ is re-created. It is possible to explore the BN Adaptor by making small changes to the JSON files, running the executable and observing the changes in the GeNIe file.

- demo BN (vulnerabilities.json has been created based on this spreadsheet)
- Under development: Praia de Faro (vulnerabilities.json has been created based on this spreadsheet)
- Under development: Zeebrugge

FEWS: GA Run Example Configurations

This section has to examples of how the above BN Adaptor can be integrated in FEWS by adjusting the GA.

(to be expanded by UAlg & IMDC)

XBeach

- General Adapter for BN using Xbeach output
- Transformation config for preprocessing for BN using Xbeach

Telemac

- General adaptor for BN using Telemac
- Transformation config for BN using Telemac

Using GeNIe

Creating a clear layout

The GeNIe files that are created by the BN Adaptor have an arbitrary layout; the variables need to be dragged in place by hand. To a get a better overview quickly:

1. Press F8 to view nodes as bar charts instead of icons
2. Navigate to Layout -> Graph Layout -> Parent Ordering and select "Top to bottom" with spacing of 20%
3. Press F5 to view the distributions
4. Select all nodes, right click on an arbitrary node to "Resize to Fit Text"

Then drag the nodes to their desired positions.

Updating distributions

Almost all nodes are so-called chance nodes meaning that they have a distribution. An exception are nodes that represent measures. These are deterministic nodes meaning that they can only be in one state at the time (with probability 100%). To select a state

- for a chance node (BC, R, H & C) double click in the state name of interest. Press F5 to update the distributions.
- for a deterministic node (M) double click on the variable name, which will open the 'Node properties' window. In the 'Definition' tab select the desired state and click OK. Press F5 to update the distributions.