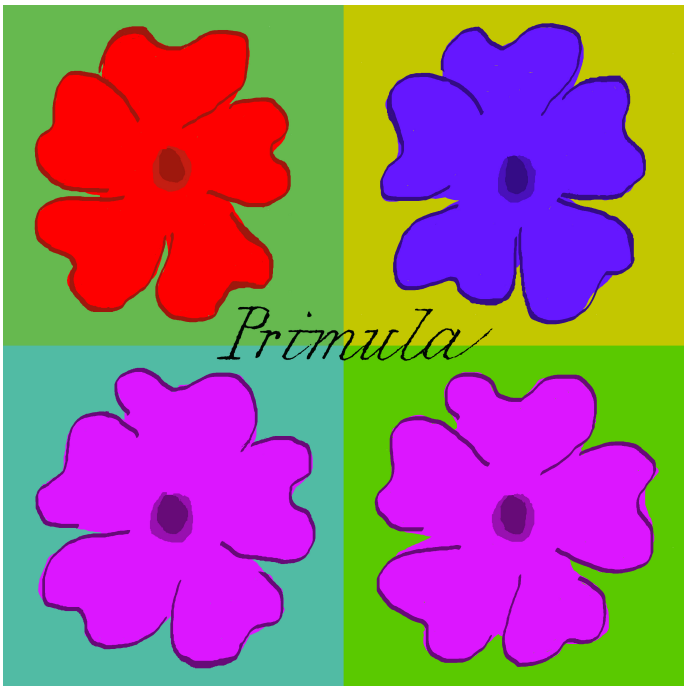
The *Primula* System: user's guide
Version 3.0
Example: Information diffusion

Manfred Jaeger
Institut for Datalogi, Aalborg Universitet, Selma Lagerløfs Vej 300, 9220 Aalborg Ø
jaeger@cs.aau.dk

Primula homepage: www.cs.aau.dk/∼jaeger/Primula

April 30, 2025



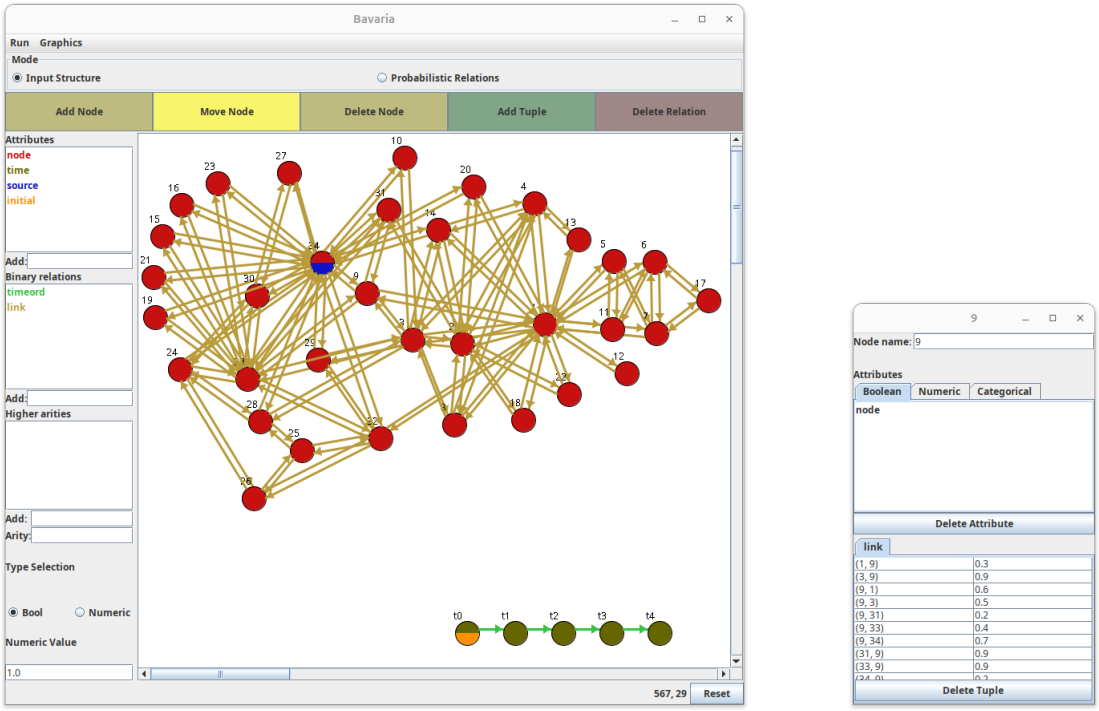


Figure 1: Input structure displayed in Bavaria.Right-clicking on a node in the Bavaria display opens a node window that displays detailed information about the node and the relations it participates in. Shown here the node window for 9.

Probabilistic Inference and Learning: Information Diffusion

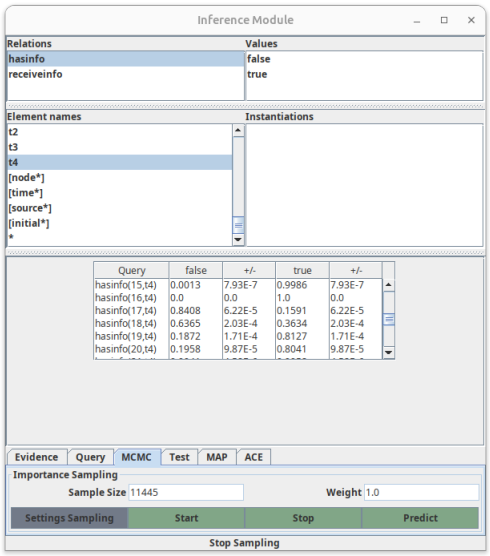
Load the model file `independent_cascade.rbn` and the data file `zachary_cascade.rdef`. The RBN model encodes the independent cascade model for information diffusion in (social) networks: at every point in time, a node can *receive* a piece of information from any neighboring node that has received the information at the previous point in time. Once a node n has received the information, the relation *hasinfo(n,t)* becomes true for all times t from this point onwards. A node can only propagate the information to its neighbors at the time point immediately after it has received the information itself. With each *link* in the network is associated a probability according to which the information will be propagated. The `zachary_cascade.rdef` file contains the Zachary Karate club social network. The usual undirected links in this network are replaced by a pair of directed links, and each link is assigned a (random) information propagation probability. Node 34 is designated a source node that is seeded with the information at time t_0. Using Modules:Bavaria the input structure can be inspected and edited (Figure 1).

Also part of the relational input structure is a discrete time line of length 5.

Inference

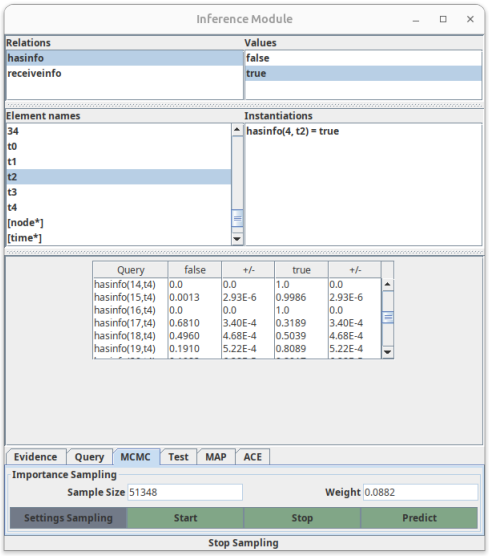
The Zachary graph already leads to a model that is too complex for exact inference. We therefore use approximate inference via the inference module:

- Open Primula:Modules:Inference Module, and select the 'Query' tab at the bottom
- Define a query by selecting 'hasinfo' in the Relations pane, and then clicking first on `[node*]` and then on `t4` in the 'Element names' pane. This creates a list of all atoms `hasinfo(n,t4)` for all nodes n, i.e., we are querying the probabilities that the information has been received at time t_4 for all nodes.
- Now select the 'MCMC' tab, and press 'Start'. This starts a Gibbs sampling process providing probability estimates that are continuously updated:



  The approximate probability that node 17 has received the information is 0.15

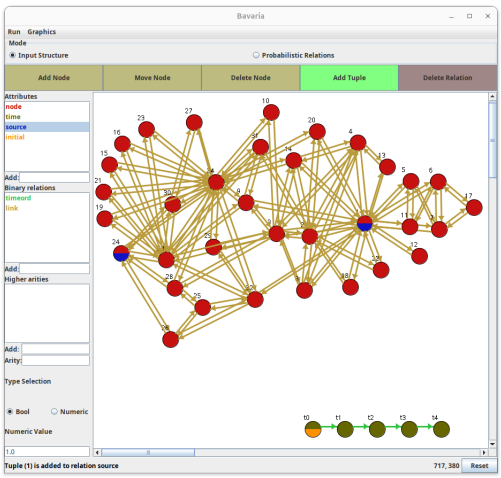
- We can condition the model on the observation that `hasinfo(4,t2)` is true: select the 'Evidence' tab, select 'hasinfo' in the 'Relations' pane and 'true' in the 'Values' pane, and click on `4` and `t2` in the 'Element names' pane. Then going back to the MCMC tab and starting a sampling process again leads to the result:



where now the probability for `hasinfo(4,t2)` has increased to 0.31.

Editing the input structure

The input structure can be modified either by editing the .rdef file, or directly in the Bavaria editor: to remove the source attribute from node 34, open its node window, select the 'source' attribute and press 'Delete Attribute'. To declare other nodes as source nodes, select the 'Add Tuple' button in Bavaria, select 'source' in the 'Attributes' list, and simply click on one or several nodes you want to declare as sources:



Learning

We can learn the propagation probabilities from observed information cascades:

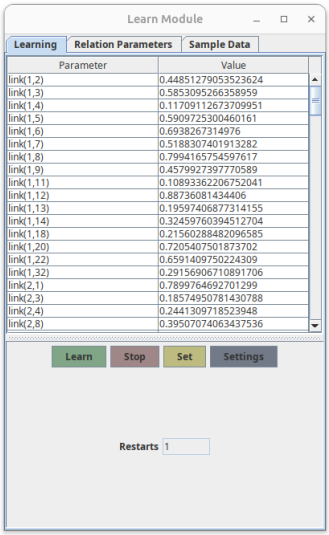


Figure 2: Learned parameters from a sample of 500 cascades. `link` values close to the source node `34` are reasonably accurate. Estimates for link values further away from `35` will be less accurate, because the data contains fewer cascades that provide relevant data for this parameter.

- Open Modules:Learn Module
- Choose the Sample Data tab, enter a sample size (=number of cascades that will be sampled), and press Sample.
- Optional: the sampled data can be saved in .rdef format by chosing Run:Save Data in the Primula console.
- In the Relation Parameters tab of the LearnModule select the `link` relation (right-click).
- The Settings menu provides a variety of different learning strategies. A simple default setting that will work in this example is to select in the Gradient Ascent:Batch tab the option Use Batch and LBFGS (options Greedy,Adagrad,Fletcher Reeves are currently not implemented/tested).
- In the Learning tab press the Learn button. The learned parameter values are displayed in the LearnModule window (Figure 2). In the settings menu one can also set a Restarts parameter. The learner will perform as many random restarts as defined by this parameter, and display the parameters with the highest likelihood found so far. Setting Restarts to -1 makes the learner to continue random restarts until interrupted by the Learn Module:Stop button.
- Pressing the Set button will set the learned relation parameter values in the current structure (overwriting the original parameters). Using Run:Save Data in the Primula console then saves the learned values to file.