GMtoolbox: Quick start
Version 1.0.1 - August, 2019

MJ. Cros, N. Peyrard, R. Sabbadin
MIAT, INRA Toulouse, France

This is a quick start to learn how to use GMtoolbox (see http://inra.fr/mia/T/GMtoolbox).

For a detailed presentation of formalization and inference in a Graphical Model (GM) with the toolbox, please refer to the user documentation of the toolbox (included in the toolbox).

1 Get GMtoolbox

From the Project Forge Files page:
`https://sourcesup.renater.fr/frs/?group_id=3868`
download the file `gmtoolbox.zip`.
Unzip the file and go to the `gmtoolbox` directory. On Linux system execute the following system commands:

```
unzip gmtoolbox.zip
cd gmtoolbox
```

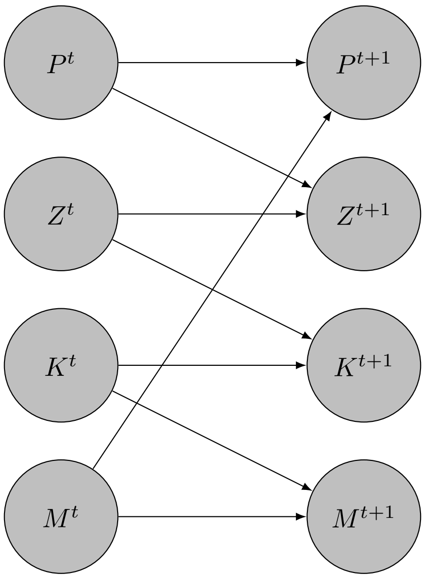
To use the toolbox in a MATLAB session, add the `gmtoolbox` directory absolute paths to MATLAB search path while being in the gmtoolbox directory. Execute the following MATLAB commands:

```
>> GMtoolbox_path = pwd;
>> addpath( genpath( GMtoolbox_path ) )
```

To access the PDF user documentation, open the file `doc/ReferenceManual.pdf` with an appropriate software.
For any question, first consult the FAQ on the GMtoolbox website:
`http://inra.fr/mia/T/GMtoolbox/download.html`
If there is no appropriate answer or if you wish to suggest a feature, add a request on the Bugs and Feature request of the project:
`https://sourcesup.renater.fr/tracker/?atid=6401&group_id=3868&func=browse`

2 A toy DBN inference ecology problem

We consider a very simplified ecological network, 4 species: plankton (P), zooplankton (Z), krill (K), marine mammals (M) which population dynamic are interdependent. We modelize the annual population dynamic by the following Dynamic Bayesian Network (DBN).

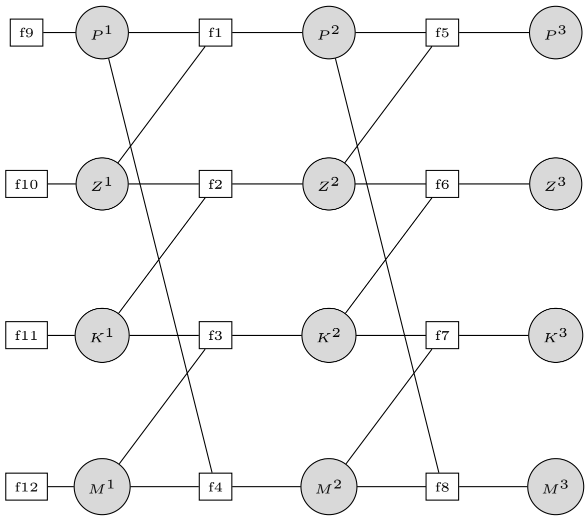


That is the population of plankton at time t+1 (P^{t+1}) is related to the population of plankton at time t (P^t) and the population of marine mammals (M^t) through feces, death... The other relations are prey-predator ones: marine mammals eat zooplankton that eat plankton.
The considering qualitative states are low (1), normal (2) and high (3) population.
The transition probabilities are simplified and are the same for all species. Considering the species V depending of also of species v, the transition $p(V^{t+1}|V^t, v^t)$ is defined by the following array T.

```
T(:,:,1) = [p1 p2 p4;   % $p(V^{t+1}=1|V^t, v^t)$
            p2 p3 p4;
            p4 p4 p4];
TT(:,:,2) =  [p4 p4 p4; % $p(V^{t+1}=2|V^t, v^t)$
              p2 p1 p2;
              p4 p4 p3];
T(:,:,3) = [p4 p4 p4;   % $p(V^{t+1}=3|V^t, v^t)$
            p4 p3 p2;
            p4 p2 p1];
```

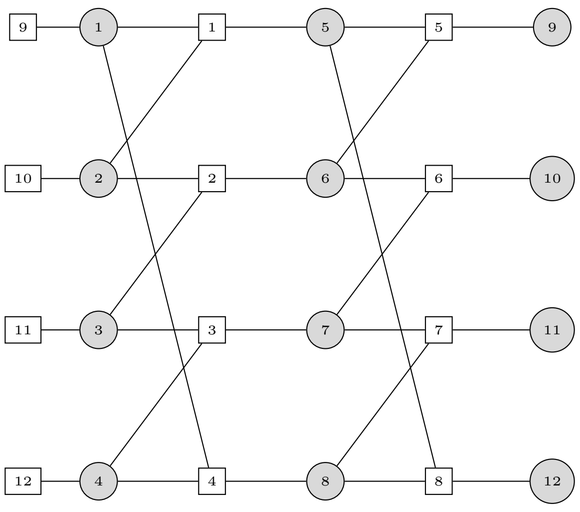
That is $p(V^{t+1} = 2|V^t = 3, v^t = 1) = T(3, 1, 2) = p4$.
For example purpose, we choose $p1 = 0.5, p2 = 0.2, p3 = 0.05, p4 = 0.01$.
Furthermore initial states are likely normal $p(V^0 = 1) = 0.25, p(V^0 = 2) = 0.5, p(V^0 = 3) = 0.25$.

We consider 3 years (2 time steps).
Lets now represent the relations for the entire graphical model with factors graph and variables.

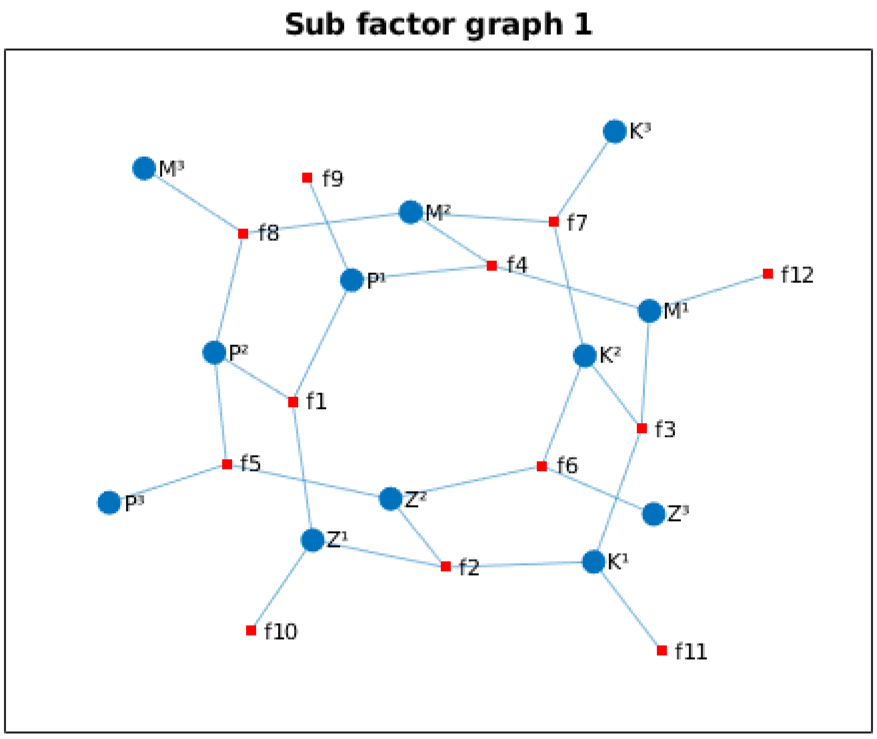


2.1 Representation of the problem

We define the list of variables as follows: $P^1, Z^1, K^1, M^1, P^2, Z^2, K^2, M^2, P^3, Z^3, K^3, M^3$. Here is the same model with the coding numbers in Matlab for variables and factors.



```
>> fg = gm_example_Ecology()
fg =
  struct with fields:
    Name: {'P¹'  'Z¹'  'K¹'  'M¹'  'P²'  'Z²'  'K²'  'M²'  'P³'  'Z³'  'K³'  'M³'}
    Card: [3 3 3 3 3 3 3 3 3 3 3 3]
     sfg: {[1×1 struct]}
>> gm_plot_fg( fg )
```



For demonstration purpose, lets check the validity of the description with the `gm_check_fg` function.

```
>> [is_OK, msg] = gm_check_fg( fg )
is_OK =
     1
msg =
    {[]}
```

No error was detected !

2.2 Inference of beliefs

If the function `gm_rg_JT` can run (for large problem memory is not sufficient), `gm_infer_GBP` provides exact values for beliefs because GBP algorithm acts as Junction tree with the generated region graph.

```
>> rg = gm_rg_JT( fg)
rg =
  1×1 cell array
    {1×1 struct}
>> [b, stop_by] = gm_infer_GBP ( fg , rg )
b =
  1×12 cell array
  Columns 1 through 4
    {3×1 double}    {3×1 double}    {3×1 double}    {3×1 double}
  Columns 5 through 8
    {3×1 double}    {3×1 double}    {3×1 double}    {3×1 double}
  Columns 9 through 12
    {3×1 double}    {3×1 double}    {3×1 double}    {3×1 double}
stop_by =
  1×1 cell array
    {'epsilon'}
>>  b{9}
ans =
    0.2012
    0.5778
    0.2210
```

Here, `gm_rg_JT` has stopped with the 'epsilon' criterium that means that the algorithm had converged. If it was not the case (stopped by 'Nmax' criterium: maximum number of iteration reached), it would be necessary to be sure that the algorithm does not oscillate and rerun it with other values for default arguments (see the second example of this QuickStart).

For information purpose, lets run the others available functions to generate region graphs. For both of them, GBP algorithm provides approximate values of beliefs.

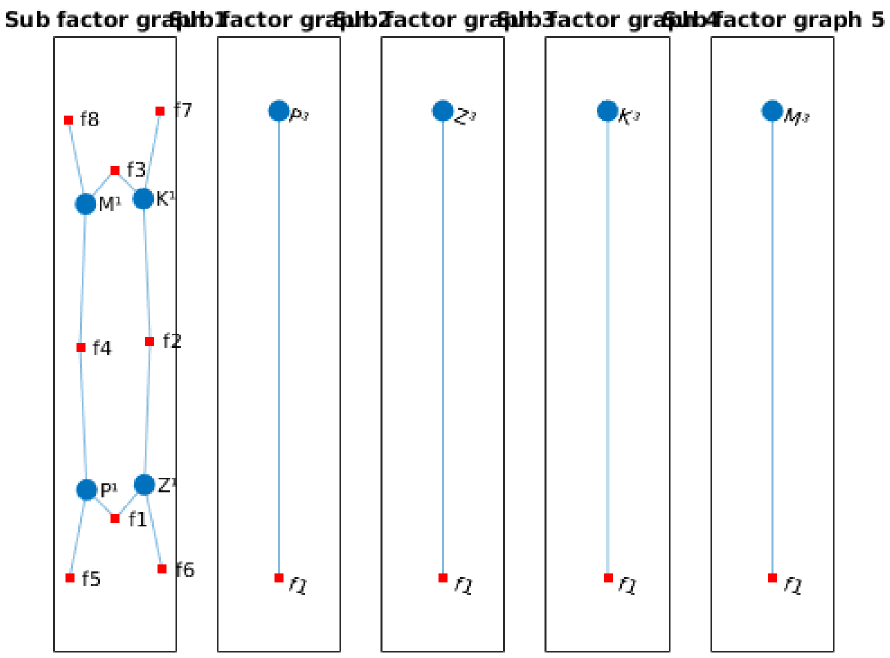
```
>> rg = gm_rg_BETHE( fg)
rg =
  1×1 cell array
    {1×1 struct}
>> [b, stop_by] = gm_infer_GBP ( fg , rg )
b =
  1×12 cell array
  Columns 1 through 4
    {3×1 double}    {3×1 double}    {3×1 double}    {3×1 double}
  Columns 5 through 8
    {3×1 double}    {3×1 double}    {3×1 double}    {3×1 double}
  Columns 9 through 12
    {3×1 double}    {3×1 double}    {3×1 double}    {3×1 double}
stop_by =
  1×1 cell array
    {'Nmax'}
>> b{9}
ans =
    0.1516
    0.6892
    0.1592

>> rg = gm_rg_CVM( fg)
rg =
  1×1 cell array
    {1×1 struct}
>> [b, stop_by] = gm_infer_GBP ( fg , rg )
b =
  1×12 cell array
  Columns 1 through 4
    {3×1 double}    {3×1 double}    {3×1 double}    {3×1 double}
  Columns 5 through 8
    {3×1 double}    {3×1 double}    {3×1 double}    {3×1 double}
  Columns 9 through 12
    {3×1 double}    {3×1 double}    {3×1 double}    {3×1 double}
stop_by =
  1×1 cell array
    {'epsilon'}
>> b{9}
ans =
    0.1517
    0.6892
    0.1591
```

We could infer that the population of plankton at time step 3 ($t = 3$), (P^3, variable 9 of the factor graph) is quite likely still normal (state 2).
Note that even for this toy example, both approximations are closed but not egal to the exact values.

2.3 Taking into account observations

Suppose that states of population are observed at time step 2 in state low: $P^2 = 1, Z^2 = 1, K^2 = 1, M^2 = 1$. Then, variables 5, 6, 7, 8 are observed at state 1.

```
>> evidence=zeros(1,length(fg.Card));
>> evidence([5 6 7 8])=1; % observation of states at the 2nd time step
>> fge = gm_include_evidence(fg, evidence)
fge =
  struct with fields:
        Name: {1×12 cell}
        Card: [3 3 3 3 3 3 3 3 3 3 3 3]
         sfg: {1×5 cell}
    evidence: [0 0 0 0 1 1 1 1 0 0 0 0]
>> gm_plot_fg( fge )
```



Taking into account these observations, the initial sub factor graph is splitted in 5 sub factor graphs.

```
>> rge = gm_rg_JT( fge);
>> [b, stop_by] = gm_infer_GBP ( fge , rge );
b =
  1×12 cell array
  Columns 1 through 5
    {3×1 double}    {3×1 double}    {3×1 double}    {3×1 double}    {3×1 double}
  Columns 6 through 10
    {3×1 double}    {3×1 double}    {3×1 double}    {3×1 double}    {3×1 double}
  Columns 11 through 12
    {3×1 double}    {3×1 double}
stop_by =
  1×5 cell array
    {'epsilon'}    {'epsilon'}    {'epsilon'}    {'epsilon'}    {'epsilon'}
>> b{9}
ans =
    0.9615
    0.0192
    0.0192
```

Now, we could infer that the population of plankton at time step 3 ($t = 3$), (P^3, variable 9 of the factor graph) is nearly for sure low (state 1).

The others ways of building region graph here provide the same values for beliefs.

3 A toy Coupled HMM epidemics problem

A Hidden Markov Model (HMM) is a Markov model in which the system being modeled is assumed to be a Markov process with unobserved (*i.e.* hidden) states. Coupled HMM (CHMM) provide interaction in state-space instead of observation states as used in classical HMM that fails to model correlation between inter-modal dependencies.

Here, a Coupled HMM is used to model the dynamics of a pest that can spread on a landscape composed of N crop fields organized in a regular grid. The neighborhood of field i, denoted N_i, is the set of the 4 closest fields (or 3 or 2, on the borders and corners of the grid).

$H_t^i \in \{0,1\}$ $(1 \leq i \leq N,\ 0 \leq t \leq T)$ is the state of crop field i at time t. State 0 (resp. 1) represents the absence (resp. presence) of the pest in the field. Variable H_t^i depends on H_{t-1}^i and of the H_{t-1}^j such that j is in N_i. The conditional probabilities of survival and apparition of the pest in field i are parameterized by 3 parameters:

- ϵ, the probability of contamination from outside the landscape (long-distance dispersal).
- ρ, the probability that the pest spreads from an infected field $j \in N_i$ $(H_t^j = 1)$ to field i between time t and $t+1$.
- ν, the probability that an infected field at time t remains infected at $t+1$

We assume that contamination events from every neighbor fields are independent. Then, if I_t^i is the number of infected neighbors of field i at time t $(I_t^i = \sum_{j \in N_i} H_t^j)$, we have

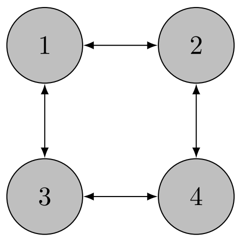
$$P(H_{t+1}^i = 1 \mid H_t^i = 0, H_t^j, j \in N_i) = \epsilon + (1-\epsilon)(1-(1-\rho)^{I_t^i})$$

and

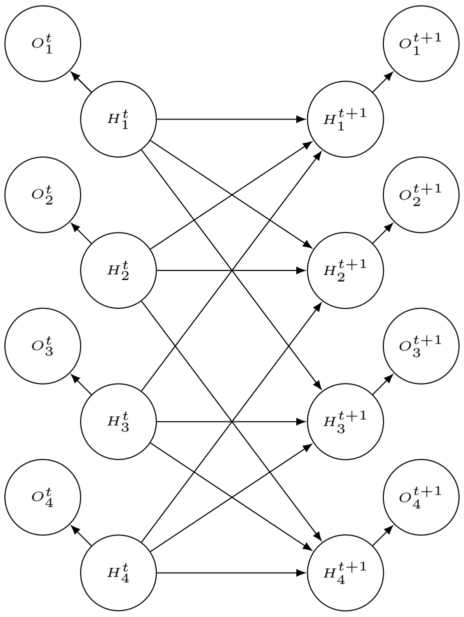
$$P(H_{t+1}^i = 1 \mid H_t^i = 1, H_t^j, j \in N_i) = \nu + (1-\nu)\left(\epsilon + (1-\epsilon)(1-(1-\rho)^{I_t^i})\right).$$

The H_t^i are hidden variables. During monitoring, a binary variable O_t^n is observed: it takes value 1 if the pest has been observed and 0 otherwise. But error of detection is possible: we can have false negative observations (the pest is there but difficult to see so it was missed) or false positive observations (the pest was mixed up with another one). We define $P(O_t^i = 1 \mid X_t^i = 0) = f_p$ and $P(O_t^i = 0 \mid X_t^i = 1) = f_n$.
"Expert" values for the dynamics parameters in the case of a weed species can be $\epsilon = 0.15, \rho = 0.2, \nu = 0.5$. For the observations distributions, we take $f_p = 0.05$ and $f_n = 0.1$.

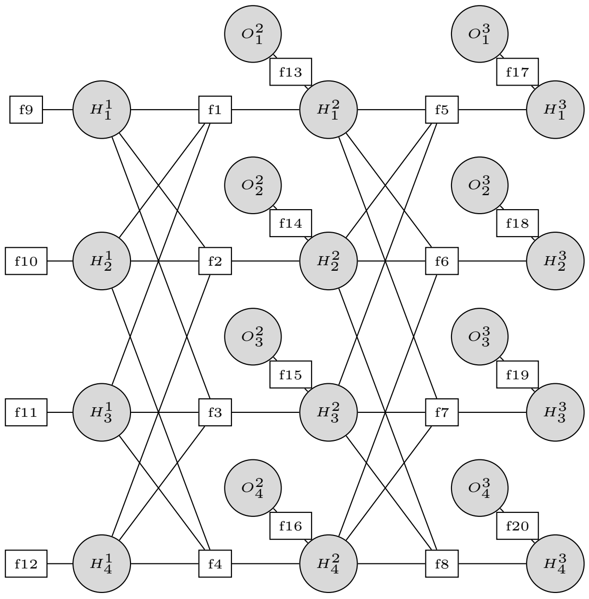
We consider a square grid of N=4 fields evolving during T=2 periods.
Lets detail the problem. Here is the set of fields and the possible propagation of pest between them.



Lets now represent state and observation variables relations between 2 time steps.



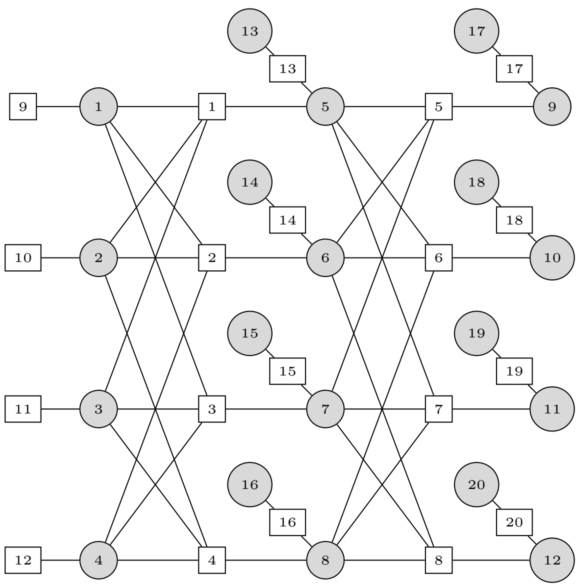
Lets represent now all the relations with factors graph and variables for the entire graphical model.



3.1 Representation of the problem

We define the list of variables as follows :
$H_1^1, H_2^1, H_3^1, H_4^1, H_1^2, H_2^2, H_3^2, H_4^2, H_1^3, H_2^3, H_3^3, H_4^3, O_1^2, O_2^2, O_3^2, O_4^2, O_1^3, O_2^3, O_3^3, O_4^3$
and we label them $1, \cdots, 20$. That is, state variables are coded $1, \cdots, 12$ and observation variables are coded $13, \cdots, 20$.

Here is the model with the coding numbers in Matlab for variables and factors.



```
>> N=4; % 4 fields
>> T=2; % 2 time steps
>> Name = {'H1^1','H2^1','H3^1','H4^1', ...
           'H1^2','H2^2','H3^2','H4^2', ...
           'H1^3','H2^3','H3^3','H4^3', ...
           'O1^2','O2^2','O3^2','O4^2', ...
           'O1^3','O2^3','O3^3','O4^3'};
```

```
>> fg = gm_example_CHMM(N, T);
>> fg.Name = Name
fg =
  struct with fields:
    Card: [2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
     sfg: {[1×1 struct]}
    Name: {1×20 cell}
```

Each variable has 2 states: 1: infected, 2: non infected.
The list of cardinalities of variables is defined in the Card attribut.

```
>> fg.Card
ans =
  Columns 1 through 20
     2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
```

We suppose that the initial states of fields are unknown. $P(H_i^1 = 1) = 0.5$ with $i = \{1, 2, 3, 4\}$
We then represent this *a priori* knowledge on initial states with factors f9, f10, f11, f12. For example, considering factor f1, the set of associated variable is H_1^1 coded 1 and the table is
$P(H_i^1 = 1) = 0.5, P(H_i^1 = 2) = 0.5$.

```
>> fg.sfg{1}.F{9}
ans =
    V: 1
    T: [2x1 double]
>> fg.sfg{1}.F{9}.T
ans =
    0.5
    0.5
```

We now define factors involved in time transitions: $f1, \cdots, f12$. Considering factor f1, the implied variables are $H_1^1, H_2^1, H_3^1, H_1^2$ which are labeled 1, 2, 3, 5.
The associated table gives $P(H_1^2 | H_1^1, H_2^1, H_3^1)$

```
>> fg.sfg{1}.F{1}.V
ans =
    1     2     3     5
>> fg.sfg{1}.F{5}.T
ans(:,:,1,1) =
    0.8500    0.6800
    0.5750    0.6600
ans(:,:,2,1) =
    0.6800    0.5440
    0.6600    0.7280
ans(:,:,1,2) =
    0.1500    0.3200
    0.4250    0.3400
ans(:,:,2,2) =
    0.3200    0.4560
    0.3400    0.2720
```

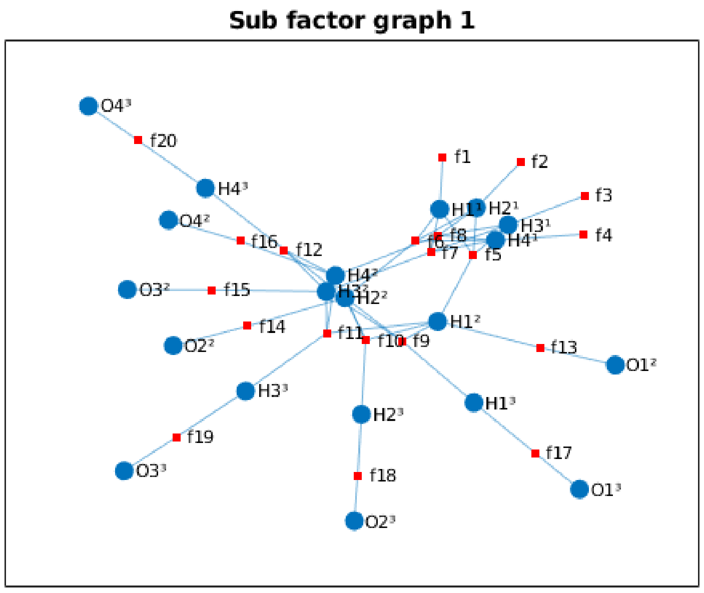
Then $P(H_1^2 = 2|H_1^1 = 1, H_2^1 = 1, H_3^1 = 1)$ is coded in T(1,1,1,2) as 0.15.

Finally, we define factors involving observation variables. Considering factor f13, it involves 2 variables O_1^2 and H_1^2 coded 5 and 13. We know that $P(H_1^2 = 1|O_1^2 = 2) = fn$ and $P(H_1^2 = 2|O_1^2 = 1) = fp$. This allows to define the associated table.

```
>> fg.sfg{1}.F{13}.V
    5    13
>> fg.sfg{1}.F{13}.T
ans =
    0.9000    0.1000
    0.0500    0.9500
```

It is possible to plot the factor graph.

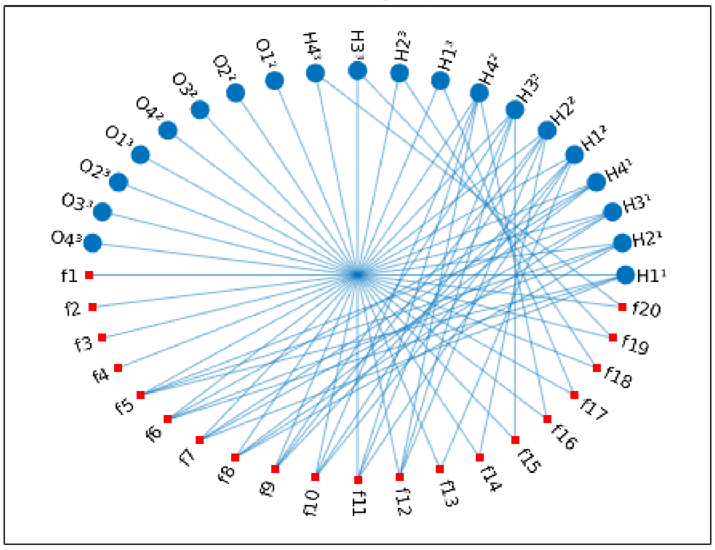
```
gm_plot_fg(fg)
```



Other layouts are possible but not really better for this problem.

```
gm_plot_fg(fg, 1, 'circle')
```

Sub factor graph 1



3.2 Inference of beliefs

If we want to know the probability of field 1 to be infected at time 3, we just have to infer it.
Before calling Generalized Belief Propagation (GBP), the fist step is to build a region graph needed by GBP. The toolbox proposes 3 functions. If the problem is simple use `gm_rg_JT`, `GBP` that will act as the Junction Tree algorithm, giving the exact probability. If not applicable, try `gm_rg_BETHE`. In this last case, GBP will act as Loopy Belief Propagation. Finally try `gm_rg_CVM` that will compute an approximate probability.

In each case, check that GBP was stopped by the 'epsilon' condition, not by 'Nmax' (the maximum number of iteration was reached). If it is the case, play with otional arguments of GBP function as in the next paragraph.

```
>> rg = gm_rg_JT(fg);
>> [b, stop_by] = gm_infer_GBP ( fg , rg );
>> stop_by
stop_by =
  1×1 cell array
    {'epsilon'}
>> b{9}(1)
ans =
    0.7062

>> rg = gm_rg_BETHE(fg);
>> [b, stop_by] = gm_infer_GBP ( fg , rg );
>> stop_by
stop_by =
  1×1 cell array
    {'epsilon'}
>>  b{9}(1)
ans =
    0.7050

>> rg = gm_rg_CVM( fg );
>> [b, stop_by] = gm_infer_GBP ( fg , rg );
>>  stop_by
stop_by =
  1×1 cell array
    {'epsilon'}
>>  b{9}(1)
ans =
    0.7050
```

The probability of field 1 at time 3 (variable H_1^3 coded 9) to be infected (coded 1) is computed with gm_rg_JT) to 0.7062. This probability is estimated to 0.7050 using gm_rg_BETHE or gm_rg_CVM.

3.3 Taking observations into account

If we only observe fields at time t=3, with just field 1 infected : $O_1^3 = 1, O_2^3 = 2, O_3^3 = 2, O_4^3 = 2$. The considered variables are coded 17, 18, 19, 20.

We just have to enter this evidence in the factor graph and infer probabilities on the new factor graph.

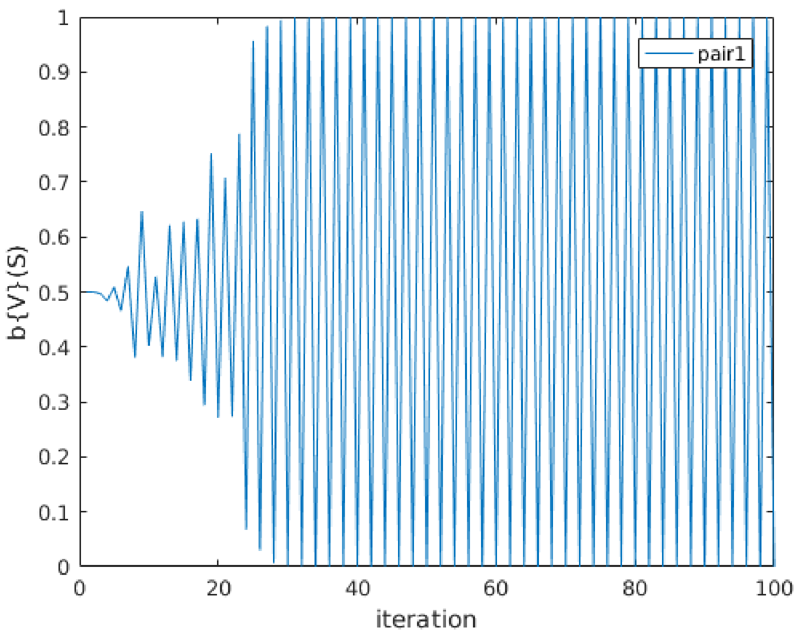
```
>> evidence = zeros(1, length(fg.Card));
>> evidence(17:20)=[1 2 2 2];

>> fge = gm_include_evidence(fg, evidence)
fge =
  struct with fields:

        Card: [2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
         sfg: {[1×1 struct]}
        Name: {1x20 cell}
    evidence: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 2 2]

>> rge = gm_rg_JT(fge);
>> [be, stop_by] = gm_infer_GBP ( fge , rge );
>> stop_by
stop_by =
  1×1 cell array
    {'epsilon'}
>>  be{1}(1)
ans =
    0.4924
```

The probability of field 1 at time 1 (variable H_1^1 coded 1) to be infected (coded 1) is computed with gm_rg_JT) to 0.4924.
For example purpose, lets approximate this probability with gm_rg_BETHE and gm_rg_CVM.

```
>> rge = gm_rg_BETHE(fge);
>> [be, stop_by] = gm_infer_GBP ( fge , rge );
>> stop_by
stop_by =
  1×1 cell array
    {'epsilon'}
>>  be{1}(1)
ans =
    0.4947

>> rge = gm_rg_CVM(fge);
>> [be, stop_by] = gm_infer_GBP ( fge , rge );
>> stop_by
stop_by =
  1×1 cell array
    {'Nmax'}
```

The probability is well estimated using gm_rg_BETHE but using gm_rg_CVM the inference algorithm is stopped by the 'Nmax' criterium (maximum number of iteration reached, by default set to 100). In this case, it is worth to evaluate the evolution of the estimation of probabilities at each iteration. For example, lets look the evolution for the probability of field 1 at time 1 (variable H_1^1 coded 1) to be infected (coded 1) for iteration from 1 to 100.

```
% We plot belief evolution
>> bt = gm_plot_belief_evolution ( 1, 1, 1, 100, fge, rge, 0.0001, 0.0 )
```



The estimation of the probability clearly oscillates. So lets try to use damping, that is make a compromize at each iteration between old and new values of messages. There is still oscillation with damp at 0.1, so we try damp at 0.2.

```
[be, stop_by] = gm_infer_GBP ( fge , rge, 100, 0.0001, 0.2, true);
n=1    delta=0.70833
n=2    delta=0.14757
n=3    delta=0.11976
n=4    delta=0.048225
n=5    delta=0.023318
n=6    delta=0.0097561
n=7    delta=0.0043376
n=8    delta=0.0020174
n=9    delta=0.00098648
n=10    delta=0.00050221
n=11    delta=0.00024589
n=12    delta=9.1312e-05
>> stop_by
stop_by =
    'epsilon'
>>  be{1}(1)
ans =
    0.4950
```

Using verbose execution (6^{th} argument of the function `gm_infer_GBP`, only 12 iterations are done for a quite good estimation of belief (0.4950 instead of 0.4924). Note that damping do not often works so well.
So, giving the observations at time 3 (with only field 1 infected), the probability of being infected at time 1 for field 1 is a little inferior at 0.5 (that is *i.e.* unknown).