GMDPtoolbox: Quick start
Version 1.1 - September, 2016

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

This is a quick start to learn how to use GMDPtoolbox.

For a detailed presentation of formalization and resolution of a GMDP problem with the toolbox, please refer to the user documentation of the toolbox (included in the toolbox).

1 Get the GMDPtoolbox

From the Project Forge Files page:
`https://mulcyber.toulouse.inra.fr/frs/?group_id=213`
download the file `gmdptoolbox.zip`.
Unzip the file and go to the `gmdptoolbox` directory. On Linux system execute the following system commands:

```
unzip gmdptoolbox.zip
cd gmdptoolbox
```

GMDPtoolbox relies on graphViz4Matlab toolbox to display graphs in functions `gmdp_see_neighborhood`, `gmdp_see_policy_graph`, `gmdp_eval_policy_value_site_contribution`. Download this toolbox, from MATLAB Central. From the web page:
`http://www.mathworks.com/matlabcentral/fileexchange/21652-graphviz4matlab`
click on the 'Download Zip' button to download the zip file in the `gmdptoolbox` directory.
Unzip the file. On Linux system execute the following system command:

```
unzip graphViz4Matlab_03Mar2010.zip
```

The toolbox relies on GraphViz available free from `http://www.graphviz.org`. On Linux, just install the graphviz package on your computer.

Note that one of the function `gmdp_linear_programming` requires the availibility of the MATLAB Optimization toolbox.
Furthermore, if MATLAB Parallel Computing Toolbox is available, the functions `gmdp_linear_programming` and `gmdp_policy_iteration_MF` are greatly speeded.
If not available, just change `parfor` in `for` in this two functions.

To use the toolbox in a MATLAB session, add the `gmdptoolbox` and `graphViz4Matlab` directory absolute paths to MATLAB search path. Execute the following MATLAB commands:

```
>> GMDPtoolbox_path = pwd;
>> addpath( GMDPtoolbox_path )
>> graphViz4Matlab_path = 'graphViz4Matlab' % give the appropriate path
>> addpath( genpath( graphViz4Matlab_path ) )
```

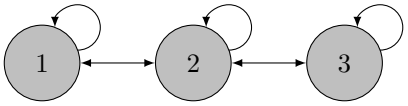
To access the PDF user documentation, open the file `doc/ReferenceManual.pdf` with an appropriate software.

For any question, first consult the FAQ on the GMDPtoolbox website:
`http://www7.inra.fr/mia/T/GMDPtoolbox/install.html#FAQ`.
If there is no appropriate answer or to suggest a feature, add a request on the Bugs and Feature request of the project:
`https://mulcyber.toulouse.inra.fr/tracker/?atid=883&group_id=213&func=browse`

2 Create a tiny GMDP epidemics management problem on 3 crop fields

Use the epidemio example provided with the toolbox to generate a problem of epidemics management on 3 crop fields. Each field can be in 2 states (1: uninfected, 2: infected). On each field, 2 actions are possible (1: normal cultural mode, 2: field fallow and treat). To have more detail (transition probabilities, rewards) on the problem, read the description of the function `gmdp_example_epidemio` in the user documentation.

For this problem the neighborhood relation corresponds to a spatial proximity between fields that enables epidemics propagation. The graph is the following:



This graph corresponds to the following adjacency matrix:

$$G = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

```
>> GMDP = gmdp_example_epidemio()
GMDP =
       M: [2 2 2]
       B: [2 2 2]
       G: [3x3 double]
    Ploc: {[4x2x2 double]  [8x2x2 double]  [4x2x2 double]}
    Rloc: {[4x2 double]  [8x2 double]  [4x2 double]}
       N: {[1 2]  [1 2 3]  [2 3]}
      VN: {[4x2 uint16]  [8x3 uint16]  [4x2 uint16]}
```

Check the validity of the description with the `gmdp_check` function.

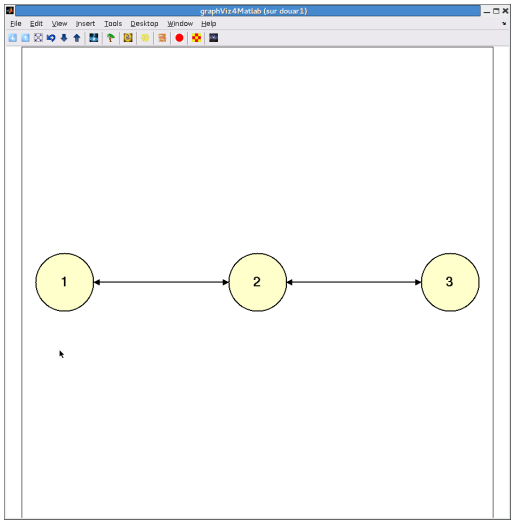
```
>> gmdp_check(GMDP)
ans =
     1
problems =
    {[]}
```

No error was detected.

The graph of neighborhood relations can be vizualized with the `gmdp_see_neighborhood` function.

```
>> gmdp_see_neighborhood(GMDP)
```

Here is the view with the circular layout. The layout may be changed with buttons of the horizontal buttons bar. The circular layout corresponds to the button with the yellow star.



Finally, define the discount factor.

```
>> discount = 0.95;
```

3 Solve the problem

To solve a GMDP problem, two algorithms (see the user documentation for references) are developed in the toolbox in two functions: `gmdp_policy_iteration_MF` and `gmdp_linear_programming`. The approximate policy iteration Mean-Field function `gmdp_policy_iteration_MF` usually provides better approximation but it is much more time consuming.

```
>> policyloc = gmdp_policy_iteration_MF(GMDP, discount)
Iteration: 1     Current approximate global value: 124.9569
Iteration: 2     Current approximate global value: 161.6177
policyloc =
    [4x1 double]    [8x1 double]    [4x1 double]
>> for i=1:3; disp( policyloc{i}'); end
     1     1     2     2
     1     1     2     2     1     1     2     2
     1     2     1     2
```

If time is an issue, and if MATLAB Optimization Toolbox is available, then use the approximate linear programming algorithm.

```
>> policyloc = gmdp_linear_programming(GMDP, discount);
Resolve site 1
Optimization terminated.
Done site 1
Resolve site 2
Optimization terminated.
Done site 2
Resolve site 3
Optimization terminated.
Done site 3
policyloc =
    [4x1 double]    [8x1 double]    [4x1 double]
>> for i=1:3; disp( policyloc{i}'); end
     1     1     2     2
     1     1     2     2     1     1     2     2
     1     2     1     2
```

The same policy is found by the two approaches.

4 Explore the policy found

Show the policy more explicitely

Display actions prescribed for each site and each neighbor's state.

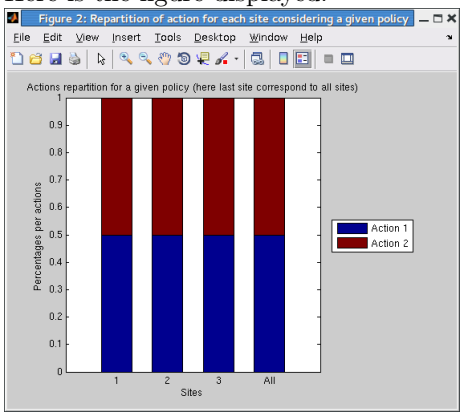
```
>> gmdp_see_policy(GMDP, policyloc)
-----------------------------------------------------------------
Site 1  Neighbor(s): 1  2
Actions: 1  1  2  2
-----------------------------------------------------------------
Site 2  Neighbor(s): 1  2  3
Actions: 1  1  2  2  1  1  2  2
-----------------------------------------------------------------
Site 3  Neighbor(s): 2  3
Actions: 1  2  1  2
```

4.1 Analyze the policy

First, we can analyse the repartition of actions prescribed by the policy (independently of sites and sites neighbors states)

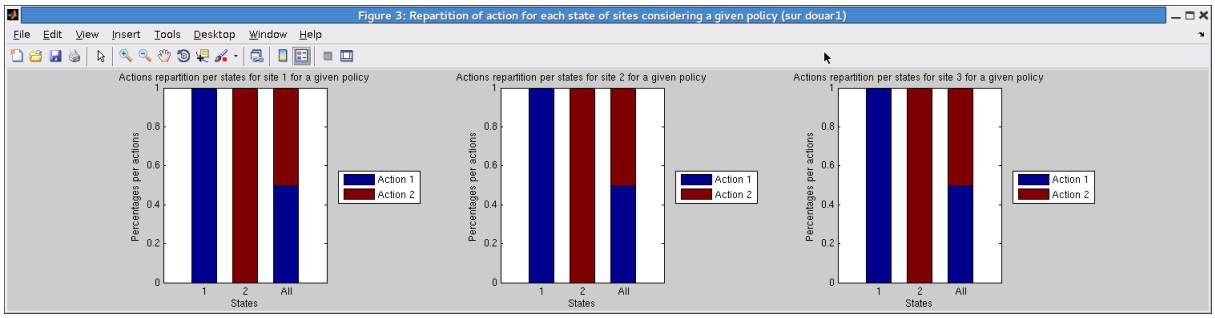
```
>> actions_repartition = gmdp_analyze_policy(GMDP, policyloc)
```

Here is the figure displayed.



The policy prescribes as many times action 1 as action 2 on each site.
Then we can analyse the repartition at particular sites, this time depending on site states (but not on site's neighbors states).

```
>> actions_repartition_state_site = gmdp_analyze_policy_neighbor(GMDP, policyloc);
```



From these repartition we can see that the policy can be expressed as follow:
If field i is uninfected (site state 1) then chose a normal cultural mode (action 1).
If field i is infected (site state 2) then leave the field fallow and treat (action 2).

4.2 Evaluate the policy with the Mean-Field approach

Compute an approximation of the global value function of the policy found.

```
>> Vloc = gmdp_eval_policy_MF (GMDP, discount, policyloc)
Vloc =
    [4x1 double]    [8x1 double]    [4x1 double]
>>for i=1:3; disp( Vloc{i}'); end
   56.7577   56.7577   53.4217   53.4217
   52.8888   52.8888   49.9880   49.9880   52.8888   52.8888   49.9880   49.9880
   56.7577   53.4217   56.7577   53.4217
```

We observe that, for sites 1, 2 and 3, reward is maximum when the site is not contaminated, whatever the neighborhood state.
Note that it is possible to have a better approximation of the value function with a better approximation of infinity (1000 instead of the default value 100) but to the cost of a larger execution time.

```
>> for i=1:n; Qloc0{i}=(ones(GMDP.M(i),1)/GMDP.M(i));end
>> Vloc = gmdp_eval_policy_MF (GMDP, discount, policyloc, Qloc0, 1000)
Vloc =
    [4x1 double]    [8x1 double]    [4x1 double]
>> for i=1:3; disp(Vloc{i}'); end
   57.0741   57.0741   53.7381   53.7381
   53.1839   53.1839   50.2831   50.2831   53.1839   53.1839   50.2831   50.2831
   57.0741   53.7381   57.0741   53.7381
```

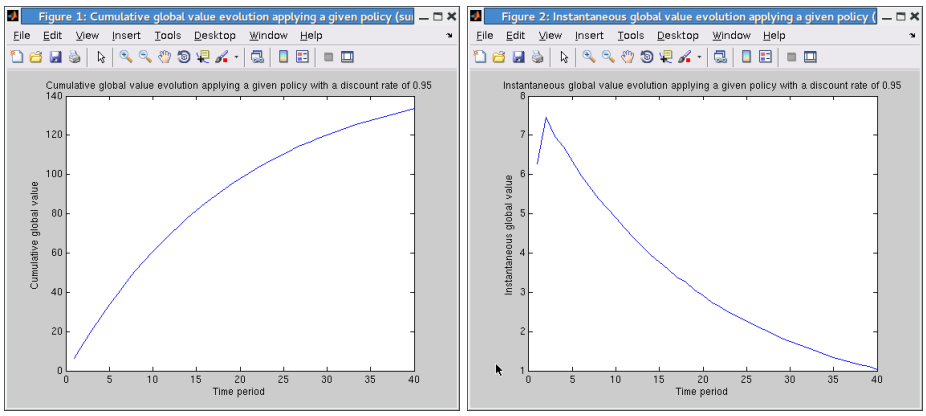
4.3 Simulate policy

Another way of evaluating the policy is to simulate the application of this policy.

```
>> [sim_state, sim_reward] = gmdp_simulate_policy(GMDP, policyloc);
```

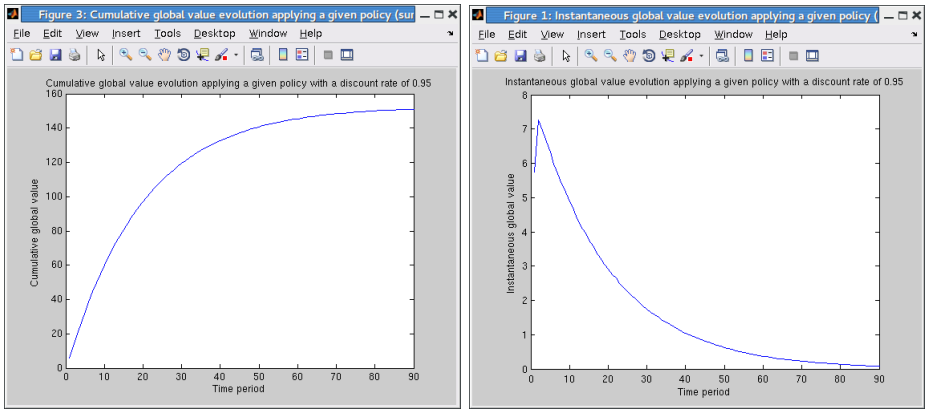
Let us now see the evolution of the global value (cumulated and instantaneous) during the 40 periods (default duration) simulated (mean value over 100 simulated trajectories).

```
>> value_evolution1 = gmdp_eval_policy_value(discount, sim_reward);
>> value_evolution2 = gmdp_eval_policy_value(discount, sim_reward, false);
```



The end of the cumulative global value is not flat and instantaneous global reward if not null after 40 periods (the default value), lets simulate more periods and re-estimate expected reward.

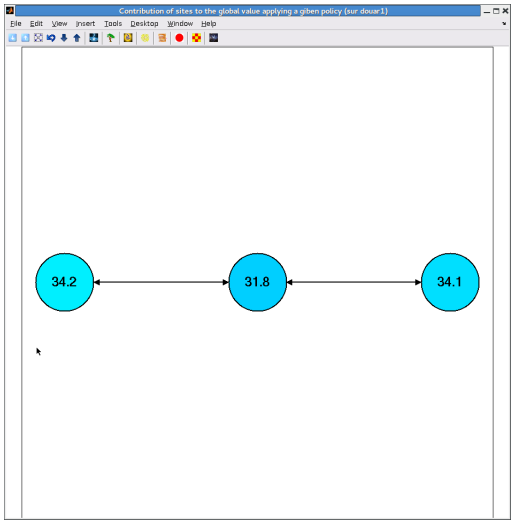
```
>> [sim_state, sim_reward] = gmdp_simulate_policy(GMDP, policyloc,100,100,90);
>> value_evolution3 = gmdp_eval_policy_value(discount, sim_reward);
>> value_evolution4 = gmdp_eval_policy_value(discount, sim_reward, false);
```



This trajectory length is more adapted to the discount factor (instantaneous reward near zero at the 90th period).
With a length of 40 periods, the cumulative global value is: value_evolution(40), 132.7156.
With a length of 90 periods, the cumulative global value is: value_evolution(90), 151.3487.
With more simulations (nb_init = 500 and nb_run = 500), we get a close approximation and find 151.0874.

The `gmdp_eval_policy_value_site_contribution` enables to quantify the contribution of each site to the cumulated global value.

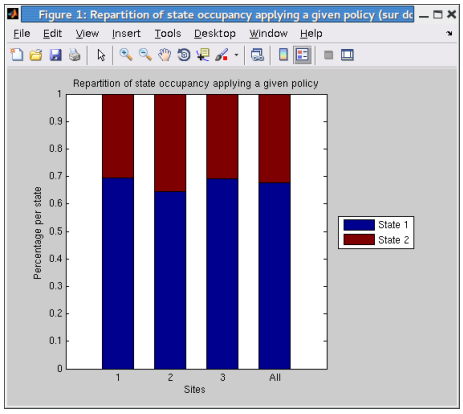
```
>> value_site = gmdp_eval_policy_value_site_contribution(GMDP, discount, sim_reward)
value_site =
   34.1663   31.7808   34.0529
```



It is quite balanced (the reason why the sites have the same color), with a slightly lower contribution of site 2.

Finally, the `gmdp_eval_policy_state_time` function enables to analyze the time spent in the different site states.

```
>> state_time = gmdp_eval_policy_state_time(GMDP, sim_state);
```



We then see that each site is nearly 30% of the time in state contaminated (state 2, colored in brown).