Graph-based Markov Decision Processes

GMDPtoolbox

Reference manual

Version 1.1 - September 2016

Marie-Josée Cros
Nathalie Peyrard
Régis Sabbadin

Contents

1 General description **3**

2 GMDP representation **5**
2.1 Description of a GMDP . . . 5
2.2 States of a site neighborhood . . . 6
2.3 Description of a policy and a value function . . . 7

3 GMDP generation functions **9**
3.1 gmdp_example_epidemio . . . 9
3.2 gmdp_example_grid . . . 11
3.3 gmdp_exemple_rand . . . 12

4 GMDP resolution functions **13**
4.1 gmdp_linear_programming . . . 13
4.2 gmdp_policy_iteration_MF . . . 14
4.3 gmdp_globalize . . . 15

5 Policy description functions **17**
5.1 gmdp_see_policy . . . 17
5.2 gmdp_see_policy_tab . . . 17
5.3 gmdp_see_policy_graph . . . 19
5.4 gmdp_analyze_policy . . . 21
5.5 gmdp_analyze_policy_neighbor . . . 22

6 Policy evaluation functions **25**
6.1 Mean-field based evaluation of a policy global value function . . . 25
6.1.1 gmdp_eval_policy_MF . . . 25
6.2 Simulation-based evaluation of a policy global value function . . . 25
6.2.1 gmdp_simulate_policy . . . 25
6.2.2 gmdp_eval_policy_value . . . 26
6.2.3 gmdp_eval_policy_value_site_contribution . . . 28
6.2.4 gmdp_eval_policy_state_time . . . 29
6.3 Evaluation of global value function . . . 31
6.3.1 gmdp_eval_policy_global_value . . . 31
6.3.2 gmdp_eval_policy_global_value_state . . . 32

7 Utilities **33**
7.1 gmdp_check_gmdp . . . 33
7.2 gmdp_check_policy . . . 34
7.3 gmdp_globalize_local_policy . . . 34
7.4 gmdp_localize_global_policy . . . 35
7.5 gmdp_see_neighborhood . . . 36

Bibliography **38**

GMDPtoolbox functions **39**

General description

Many problems of management of systems in complex domains such that ecology, agricultural production, finance, robotics, can be described as a sequential decision under uncertainty problem where an agent has to take a succession of decisions without a precise and deterministic knowledge of the effects of the decisions on the system.

A classical framework of modeling: MDP

The Markov Decision Processes (MDP) framework [1, 2] is a classical approach to model and solve such problems of sequential decisions under uncertainty.

In this framework, the state of the system defines the situation of the agent and the environment at each moment. The dynamic of the system is a result of the interaction between the actions taken by the agent, the own dynamic of the environment and the dynamic of the agent. This dynamic is non-deterministic (the same causes do not implies the same effects). It is described by a stochastic transition function describing a probability distribution on the states at time $t + 1$ for all possible pairs state - action at time t.

When the agent executes an action, he receives an instantaneous reward. The agent chooses actions in order to maximize a satisfaction criterion related to the reward obtained with the followed trajectory (sequence of pairs state - action).
A policy models the agent decisional behavior. It is a function that associates to a state the action to execute.

Solving a MDP problem is finding a policy that maximizes a criterion that expresses the expected sum of instantaneous rewards. Various algorithms exists to solve this problem. However, in practice, the effective resolution is only possible for limited system states or actions size.

A new framework: GMDP

To be able to solve large problems, different frameworks of factored MDP [3, 4] were defined to represent problems in a compact way.
More precisely, when the states and action spaces are multidimensional, the Graph-based Markov Decision Processes (GMDP) framework [5, 6] has been defined. In this case, the transition and reward functions are decomposed into local functions and the dependence relations between variables can be represented by a graph. The determination of a global optimal policy is still unattainable so approximated algorithms [5, 6] have been developed to solve GMDP problems.
GMDP are particularly well adapted to model spatial decision problems. In these problems, each variable and each action are attached to a spatial location, called a site, represented by a node of the graph.

A GMDP toolbox

The MATLAB toolbox called GMDPtoolbox provides the main resolution algorithms for GMDP, with tools for describing and analyzing policies output of these algorithms. The toolbox is presented

in a web site [7]. The code under BSD license is available on the Software Forge of the project [8].

To help handling the toolbox, a quick start guide describes how to install GMDPtoolbox and graphViz4matlab toolbox used to display graphs. It also detailed step by step the resolution of a tiny epidemics management problem on 3 crop fields.

GMDP representation

Description of a GMDP

A discrete-time GMDP is defined by a 5-tuple $< S, A, N, p, r >$ where :

- S is the state space,
  $S = S_1 \times \cdots \times S_n$ with S_i the finite state space of site i
- A is the action space,
  $A = A_1 \times \cdots \times A_n$ with A_i the finite action space of site i
- N is the set of sites neighborhood relations,
  $N = \{N_i,\ \forall i = 1, \ldots, n\}$ with N_i, the set of neighbors of site i
- p is the set of local sites transition probability functions,
  $p = \{p_i(s'_i|s_{N_i}, a_i),\ \forall i = 1, \ldots, n,\ \forall s'_i, s_{N_i}, a_i\}$
  with $p_i(s'_i|s_{N_i}, a_i)$ the probability for site i of being in state s'_i at time $t+1$ given that at time t the neighborhood of the site is in state s_{N_i} and action a_i is performed.
- r is the set of local sites reward functions
  $r = \{r_i(s_{N_i}, a_i),\ \forall i = 1, \ldots, n,\ \forall s_{N_i},\ \forall a_i\}$
  with r_i the reward obtained from site i at time t when the neighborhood of site i is in state s_{N_i} and action a_i is performed.

In GMDPtoolbox, a problem is represented in a data structure (called GMDP in this documentation) which is a MATLAB structure with the following attributes:

- M : vector (1xn) of number of states per site,
- B : vector (1xn) of number of actions per site,
- G : adjacency matrix (nxn), column i describes N_i,
- Ploc : cell array (1xn) of site transition probability functions stored, for site i, in an array of dimension
$$(\prod_{k \in N_i} |\mathcal{S}_k|) \times |\mathcal{S}_i| \times |\mathcal{A}_i|$$
- Rloc : cell array (1xn) of site reward functions stored, for site i, in a matrix of dimension
$$(\prod_{k \in N_i} |\mathcal{S}_k|) \times |\mathcal{A}_i|$$

For the simple epidemics management problem with 3 fields (sites) provided in the toolbox (see description in paragraph 3.1), the structure is instantiated as follow.

```
>> % Generate a GMDP for a simple epidemics management problem with 3 fields
>> 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]}
>> % Let see the neighbor relations with the adjacency matrix
>> GMDP.G
ans =
     1     1     0
     1     1     1
     0     1     1
```

Here, the neighbors of field 1 are field 1 itself and field 2.

States of a site neighborhood

Considering a site, the state of the neighborhood is defined using the lexicographic order considering the neighbors in the numerical growing order.
For example, for a neighborhood containing 3 sites $\{2, 4, 5\}$ with respectively 2, 3, 2 possible states, here is the numbering of the neighborhood.

s_2	s_4	s_5	$s_{\{2,4,5\}}$
1	1	1	1
1	1	2	2
1	2	1	3
1	2	2	4
1	3	1	5
1	3	2	6
2	1	1	7
2	1	2	8
2	2	1	9
2	2	2	10
2	3	1	11
2	3	2	12

For example, if the neighborhood is in state 7 ($s_{\{2,4,5\}} = 7$), that means that site 2 is in state 2 and sites 4 and 5 are in state 1. Note that the lexicogrpahic order is also used to convert a vector of one action per site into a single number.
To speed up computation, 2 attributes are pre-computed from M and G and added to the representation of a GMDP problem:

- N : cell array (1xn) of sites lists of neighbors (vector 1 x $|N_i|$),
- VN : cell array (1xn) of sites tables of detailed state of neighbors (matrix $\prod_{k \in N_i} |S_k|$ x $|N_i|$).

These 2 attributes of a GMDP structure are computed by the functions that generate GMDP problems and by the function that checks others attributes, as illustrated below.

```
>> % manual creation of a GMDP
>> GMDP.M=[2 2]; GMDP.B=[2 2]; GMDP.G=ones(2,2);
>> GMDP.Ploc{1}(:,:,1)=repmat([0 1],4,1); GMDP.Ploc{1}(:,:,2)=repmat([0 1],4,1);
>> GMDP.Ploc{2}=GMDP.Ploc{1};
>> GMDP.Rloc{1}=rand(4,2); GMDP.Rloc{2}=rand(4,2);
>> % checking that the structure is correct
>> [isOK, problems, GMDP] = gmdp_check_gmdp(GMDP); GMDP
GMDP =
       M: [2 2]
       B: [2 2]
       G: [2x2 double]
    Ploc: {[4x2x2 double]  [4x2x2 double]}
    Rloc: {[4x2 double]  [4x2 double]}
       N: {[1 2]  [1 2]}
      VN: {[4x2 uint16]  [4x2 uint16]}
```

Description of a policy and a value function

In the GMDP framework, a policy is a set of site (or local) policies.

$$\pi = \{\pi_i\}_{i=1,\dots,n} \text{ with } \pi_i : S_{N_i} \mapsto A_i$$

In the GMDPtoolbox, a policy is described by a cell array (1xn) of site policies, stored for site i in a vector of length

$$(\prod_{k \in N_i} |\mathcal{S}_k|).$$

Considering the simple epidemics management problem with 3 fields (sites) provided by the toolbox (see description in paragraph 3.1), a policy is instantiated as follow.

```
>> % Generate a GMDP for a simple epidemics management problem with 3 sites
>> GMDP = gmdp_example_epidemio();
>> % Find an approximate optimal policy
>> policyloc = gmdp_policy_iteration_MF(GMDP, 0.9)
Iteration: 1     Current approximate global value: 65.4101
Iteration: 2     Current approximate global value: 80.0669
policyloc =
    [4x1 double]    [8x1 double]    [4x1 double]
>> % Display the policy (one line per field)
>> 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 first displayed line of policyloc (that is policyloc{1}), expresses the policy for field 1: a recommended action for each of the 4 neighborhood possible states. For example when field 1 neighborhood is in state 3 (that is field 1 in state 2 and field 2 in state 1), action 2 (that is policyloc{1}(3)) is recommended on field 1.
If all sites are in state 1 (which correspond to the global state [111] encoded as 1), then all the sites neighborhoods are in state 1 and the policy recommends action 1 for all sites. So, the global action is [111], encoded as 1.

In the GMDP framework, the global value function of a policy for a infinite horizon problem is defined as for a standard MDP:

$$V^\pi(s) = E[\sum_{t=0}^{\infty} \gamma^t r^t]$$

The global value function is approximated by a sum of local value functions $\sum_{i=1}^{n} V_i^\pi(s_{N_i})$, therefore it can be described by the set $\{V_i^\pi(s_{N_i}),\ \forall i,\ \forall s_{N_i}\}$ of local value functions which has the same structure as a policy.

```
>> % Approximate the value function
>> Vloc = gmdp_eval_policy_MF (GMDP, 0.9, policyloc)
Vloc =
    [4x1 double]    [8x1 double]    [4x1 double]
>> % Display the value function (one line per field)
>> for i=1:3; disp(Vloc{i}');end
   28.9641   28.9641   25.5974   25.5974
   26.9800   26.9800   24.0310   24.0310   26.9800   26.9800   24.0310   24.0310
   28.9641   25.5974   28.9641   25.5974
```

For example, if the system is in state 1 (that is all sites in state 1), all the sites neighbors are in state 1 and the approximate global value is the sum of the local (on a site) values: $Vloc1(1) + Vloc2(1) + Vloc3(1) = 28.9641 + 26.9800 + 28.9641 = 84.9082$.

GMDP generation functions

gmdp_example_epidemio

Description

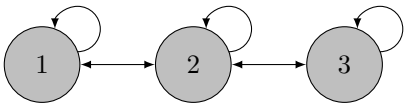
Generates a tiny GMDP epidemics management problem on 3 crop fields.
Each crop field can be in two states ($|S_i| = 2, \forall i = 1, 2, 3$):

- uninfected (coded 1) or
- infected (coded 2).

Each field is susceptible to contamination by a pathogen. Contamination can either be long range, or over a short distance between neighboring fields. When a field is contaminated, the yield decreases. Decisions about crop field management are taken each year and two actions can be applied to each field ($|S_i| = 2, \forall i = 1, 2, 3$):

- a normal cultural mode (coded 1) or
- the field is left fallow and treated (coded 2).

The problem is to optimize the choice of a long-term policy in term of expected yield. The topology of the area is represented by a graph. There is one node per crop field. A directed edge between two nodes represents potential contamination. The neighborhood relationship is symmetric since we assume that infection can spread in any direction: $N_1 = \{1, 2\}, N_2 = \{1, 2, 3\}, N_3 = \{2, 3\}$. Here is the graph for the 3 considered fields.



Transition probabilities are parametrized by the following variables:

- the probability pd that a field infected becomes uninfected when it is left fallow and treated,
- the probability $p\epsilon$ of long-distance contamination,
- the probability pc that a field be contaminated by an infected neighboring field,

The probability $p(m_i)$ that a field with m_i infected neighboring fields moves from state uninfected to infected under a normal cultural mode is defined by:

$$p(m_i) = p\epsilon + (1 - p\epsilon)(1 - (1 - pc)^{m_i})$$

For a site i in state s_i at time t and s'_i at time $t + 1$, the following table summarizes the probabilities of transition when action a_i is performed.

	$a_i = 1$		$a_i = 2$	
	$s'_i = 1$	$s'_i = 2$	$s'_i = 1$	$s'_i = 2$
$s_i = 1$	$1 - p(m_i)$	$p(m_i)$	1	0
$s_i = 2$	0	1	pd	$1 - pd$

Harvest is affected by the state of the crop field and the chosen action. The maximal yield (noted r) can be achieved for an uninfected field with normal treatment. If the field is contaminated, the reward is only r/2. Otherwise, a field left fallow and treated produces no reward. For a filed i in state s_i at time t, the following table summarizes the rewards obtained when action a_i is performed.

	$a_i = 1$	$a_i = 2$
$s_i = 1$	r	0
$s_i = 2$	$r/2$	0

Syntax

```
GMDP = gmdp_example_epidemio()
GMDP = gmdp_example_epidemio(pepsilon)
GMDP = gmdp_example_epidemio(pepsilon, pc)
GMDP = gmdp_example_epidemio(pepsilon, pc, pd)
GMDP = gmdp_example_epidemio(pepsilon, pc, pd, r)
```

Arguments

- **pepsilon** : probability of long distance contamination (default value: 0.2)
- **pc** : probability that a field be contaminated by an infected neighborhood field by an infected neighborhood field (default value: 0.5)
- **pd** : probability that a field infected becomes uninfected when it is left fallow and treated (default value: 0.8)
- **r** : reward if state is not contaminated and usual action is performed (default value: 4)

Evaluation

- **GMDP** : structure (see paragraph 2.1) defining the problem

Example

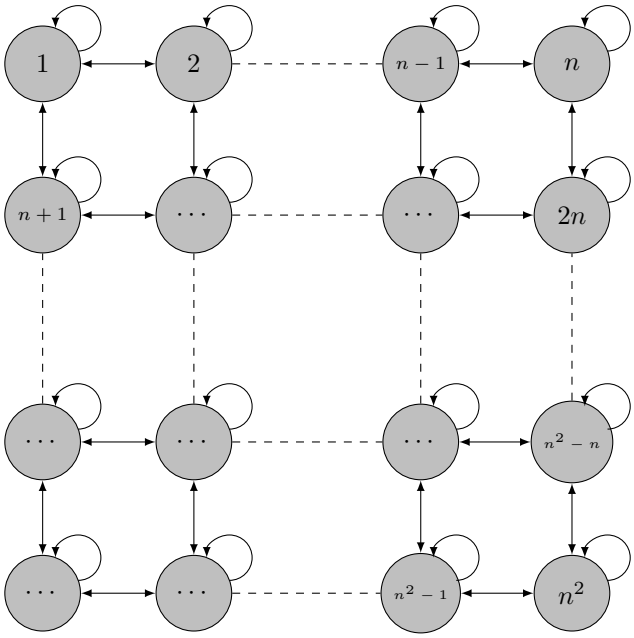
```
>> GMDP = gmdp_example_epidemio(0.5)
GMDP =
       M: [2 2 2]
       B: [2 2 2]
       G: [3x3 double]
    Ploc: {[4x2x2 double]  [8x2x2 double]  [4x2x2 double]}
    Rloc: {[4x2 double]  [8x2 double]  [4x2 double]}
>>% Let examine the transition probabilities for field 1
>>GMDP.Ploc{1}
ans(:,:,1) =
    0.5000    0.5000
    0.2500    0.7500
         0    1.0000
         0    1.0000
ans(:,:,2) =
    1.0000         0
    1.0000         0
    0.8000    0.2000
    0.8000    0.2000
```

Field 1 neighborhood has 4 possible states; 2 actions are available.
For field 1 neighborhood in state 3 (that is field 1 in state 2 and field 2 in state 1) if performing action 2 then the probability for field 1 to evolve to state 1 is 0.8 (GMDP.Ploc{1}(3,1,2)).

gmdp_example_grid

Description

Generates a random GMDP problem with a grid nxn for neighbor sites relations.
The figure below displays sites indices and neighborhood relationships for the case $n = 5$



Each site has 2 possible states. Each site has 2 possible actions. Transition probabilities and rewards are randomly set.

Syntax

```
GMDP = gmdp_example_grid(n)
```

Arguments

- **n** : number of sites per row (or column) of the square grid of sites.

Evaluation

- **GMDP** : structure (see paragraph 2.1) defining the problem

Example

```
>> GMDP = gmdp_example_grid(5)
GMDP =
      M: [2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
      B: [2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
      G: [25x25 double]
   Ploc: {1x25 cell}
   Rloc: {1x25 cell}
      N: {1x25 cell}
     VN: {1x25 cell}
```

gmdp_exemple_rand

Description

Generates a random GMDP problem.

Syntax

```
GMDP = gmdp_example_rand()
GMDP = gmdp_example_rand(n)
GMDP = gmdp_example_rand(n, M)
GMDP = gmdp_example_rand(n, M, B)
GMDP = gmdp_example_rand(n, M, B, maxN)
GMDP = gmdp_example_rand(n, M, B, maxN, G)
```

Arguments

- **n** : number of sites (default value: 3)
- **M** : vector (1xn) of numbers of states for each site (by default 2 states by site)
- **B** : vector (1xn) of numbers of actions for each site (by default 2 states by site)
- **maxN** : maximum number of neighbors for a site (by default $min(n, 4)$)
- **G** : adjacency matrix describing the neighborhood of each site. G is randomly generated such that All sites have at least one other site in their neighborhood and neighborhood size is at most maxN

Evaluation

- **GMDP** : structure (see paragraph 2.1) defining the problem

Example

```
>> rng(0);GMDP = gmdp_example_rand(4,[2 3 4 5],[5 4 3 2])
GMDP =
       M: [2 3 4 5]
       B: [5 4 3 2]
       G: [4x4 double]
    Ploc: {[120x2x5 double]  [5x3x4 double]  [60x4x3 double]  [60x5x2 double]}
    Rloc: {[120x5 double]  [5x4 double]  [60x3 double]  [60x2 double]}
       N: {[1 2 3 4]  [4]  [2 3 4]  [2 3 4]}
      VN: {[120x4 uint16]  [5x1 uint16]  [60x3 uint16]  [60x3 uint16]}
```

GMDP resolution functions

Solving a GMDP problem means computing the policy with largest global value. In the following we will describe functions that provide approximate resolution of a GMDP problem, they are functions that rely on approximations of the global value function by local value functions, to provide a good (but not optimal) policy.
The approximate policy iteration Mean-Field function `gmdp_policy_iteration_MF` usually provides better approximation but it is much more time consuming. Just to give an idea , on a personal computer HP Z420 workstation with 16 Go of RAM, it is possible to create a GMDP of 10000 variables with the gmdp_example_rand function in about 3 mn, the resolution with the gmdp_policy_iteration_MF function requires less than 3 h whereas the resolution with the gmdp_linear_programmi is achieved in less than 2 mn.

gmdp_linear_programming

Description

Approximate resolution of a GMDP problem with an approximated linear programming algorithm (MATLAB Optimization Toolbox is required).

Syntax

```
policyloc = gmdp_linear_programming (GMDP, discount)
policyloc = gmdp_linear_programming (GMDP, discount, options)
```

Arguments

- **GMDP** : structure (see paragraph 2.1) defining the problem
- **discount** : discount rate in]0; 1[
- **options** : define options for MATLAB linprog function (see MATLAB documentation, `http://fr.mathw`

Evaluation

- **policyloc** : approximate optimal policy (see paragraph 2.3 for a description of the structure)

Example

```
>> GMDP = gmdp_example_epidemio();
>> options = optimoptions('linprog','Algorithm','dual-simplex','MaxTime',1200);
>> policyloc = gmdp_linear_programming (GMDP, 0.9)
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]
```

gmdp_policy_iteration_MF

Description

Approximate resolution of a GMDP problem with the Mean-Field approximated policy iteration algorithm. This function is iterative and involves calls to the function gmdp_eval_policy_MF (see subsection 6.1.1) to evaluate the current policy value. Therefore some of the optional arguments of gmdp_policy_iteration_MF are required for gmdp_eval_policy_MF.
May use MATLAB Parallel Computing Toolbox if available.

Syntax

```
[policyloc, iter] = gmdp_policy_iteration_MF(GMDP,discount)
[policyloc, iter] = gmdp_policy_iteration_MF(GMDP,discount, max_iter)
[policyloc, iter] = gmdp_policy_iteration_MF(GMDP,discount, max_iter, policyloc0)
[policyloc, iter] = gmdp_policy_iteration_MF(GMDP,discount, max_iter, policyloc0, Qloc0)
[policyloc, iter] = gmdp_policy_iteration_MF(GMDP,discount, max_iter, policyloc0, Qloc0,
                            Tend_eval)
[policyloc, iter] = gmdp_policy_iteration_MF(GMDP,discount, max_iter, policyloc0, Qloc0,
                            Tend_eval, max_iter_update)
```

Arguments

- **GMDP** : structure (see paragraph 2.1) defining the problem
- **discount** : discount rate in]0; 1[
- **max_iter** : maximum number of iterations to be done (default value: 100)
- **policyloc0** : an initial policy (default action 1 in all states), its structure is described in subsection 2.3
- **Qloc0** : proposal probability distributions for sites states at $t = 0$ (default value uniform for each site)
  Qloc0 is a cell array (1xn) of site state probability which is stored for site i in a vector of length $|\mathcal{S}_i|$.
- **Tend_eval** : approximation of infinity (default value: 100)
- **max_iter_update** : maximum number of iterations to update current policy (default value: 10)

Evaluation

- **policyloc** : approximate optimal policy (see paragraph 2.3 for a description of the structure)
- **Vloc** : set of local value functions (see paragraph 2.3 for a description of the structure) defining the approximate global value of policyloc
- **iter** : number of iterations.

Example

```
>> GMDP = gmdp_example_epidemio();
>> [policyloc, Vloc, iter] = gmdp_policy_iteration_MF(GMDP, 0.9)
Iteration: 1     Current approximate global value: 65.4101
Iteration: 2     Current approximate global value: 80.0669
policyloc =
    [4x1 double]    [8x1 double]    [4x1 double]
Vloc =
    [4x1 double]    [8x1 double]    [4x1 double]
iter =
     2
```

gmdp_globalize

Description

Translates a GMDP problem into a MDP problem in order to use MDPtoolbox [9] functions for the resolution.

Syntax

```
[P, R] = gmdp_globalize (GMDP)
```

Argument

- **GMDP** : structure (see paragraph 2.1) defining a problem

Evaluation

- **P** : a transition probability array of dimension

$$(\prod_{k\in 1\ldots n} |\mathcal{S}_k|) \times (\prod_{k\in 1\ldots n} |\mathcal{S}_k|) \times (\prod_{k\in 1\ldots n} |\mathcal{A}_k|)$$

- **R** : a reward matrix of dimension

$$(\prod_{k\in 1\ldots n} |\mathcal{S}_k|) \times (\prod_{k\in 1\ldots n} |\mathcal{A}_k|)$$

Example

```
>> rng(1)
>> GMDP = gmdp_example_rand(2, [2 2], [2 2], 2, [0 1;1 0])
GMDP =
       M: [2 2]
       B: [2 2]
       G: [2x2 double]
    Ploc: {[2x2x2 double]  [2x2x2 double]}
    Rloc: {[2x2 double]  [2x2 double]}
       N: {[2]  [1]}
      VN: {[2x1 uint16]  [2x1 uint16]}
>> [P, R] = gmdp_globalize (GMDP)
P(:,:,1) =
    0.8816    0.1181    0.0002    0.0000
    0.6212    0.0832    0.2607    0.0349
    0.5669    0.4328    0.0002    0.0001
    0.3994    0.3050    0.1676    0.1280
P(:,:,2) =
    0.7481    0.2517    0.0002    0.0001
    0.5271    0.1773    0.2212    0.0744
    0.7380    0.2617    0.0002    0.0001
    0.5200    0.1844    0.2182    0.0774
P(:,:,3) =
    0.3886    0.0521    0.4932    0.0661
    0.1860    0.0249    0.6959    0.0932
    0.2499    0.1908    0.3172    0.2422
    0.1196    0.0913    0.4475    0.3417
P(:,:,4) =
    0.3298    0.1109    0.4185    0.1408
    0.1578    0.0531    0.5905    0.1986
    0.3253    0.1154    0.4129    0.1464
    0.1557    0.0552    0.5826    0.2066
R =
    1.1975    0.7102    1.2199    0.7326
    1.3396    0.8522    1.4860    0.9986
    1.3650    1.0891    1.3875    1.1115
    1.5071    1.2311    1.6535    1.3775
```

Policy description functions

gmdp_see_policy

Description

Displays actions prescribed by a policy.

Syntax

```
gmdp_see_policy(GMDP, policyloc)
```

Arguments

- **GMDP** : structure (see paragraph 2.1) defining the problem
- **policyloc** : a policy (see paragraph 2.3 for a description of the structure)

Evaluation

- Displays information in the command window.

Example

```
>> GMDP = gmdp_example_epidemio();
>> policyloc = gmdp_linear_programming(GMDP,0.9);
>> 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
```

gmdp_see_policy_tab

Description

Displays the tables of the actions prescribed by a policy with explicit specification of all neighbors states.

Syntax

```
policyloc_verbose = gmdp_see_policy_tab(GMDP, policyloc)
policyloc_verbose = gmdp_see_policy_tab(GMDP, policyloc, sites)
policyloc_verbose = gmdp_see_policy_tab(GMDP, policyloc, sites, isdisplayed)
```

Arguments

- **GMDP** : structure (see paragraph 2.1) defining the problem
- **policyloc** : a policy (see paragraph 2.3 for a description of the structure)
- **sites** : vector of sites to be considered (default value: all sites)
- **isdisplayed** : if true: displays verbose policy, if false: no display (default value: true)

Evaluation

- **policyloc_verbose** : policy with detailed neighborhood state for each site; cell array (1xn) containing a matrix (nb of neighborhood state)x(nb of neighbor +1). The first column of a matrix contains the actions prescribed by the policy for each neighborhood state.
  The last column(s) contains the state of the neighborhood as a vector.

Example

```
>> GMDP = gmdp_example_epidemio();
>> policyloc = gmdp_linear_programming(GMDP,0.9);
>> policyloc_verbose = gmdp_see_policy_tab(GMDP, policyloc)
-----------------------------------------------------------------
Site 1  Neighbor(s): 1  2
-----------------------------------------------------------------
Action | State | State of the neighbor(s)
1      | 1     | 1  1
1      | 2     | 1  2
2      | 3     | 2  1
2      | 4     | 2  2
-----------------------------------------------------------------
Site 2  Neighbor(s): 1  2  3
-----------------------------------------------------------------
Action | State | State of the neighbor(s)
1      | 1     | 1  1  1
1      | 2     | 1  1  2
2      | 3     | 1  2  1
2      | 4     | 1  2  2
1      | 5     | 2  1  1
1      | 6     | 2  1  2
2      | 7     | 2  2  1
2      | 8     | 2  2  2
-----------------------------------------------------------------
Site 3  Neighbor(s): 2  3
-----------------------------------------------------------------
Action | State | State of the neighbor(s)
1      | 1     | 1  1
2      | 2     | 1  2
1      | 3     | 2  1
2      | 4     | 2  2
policyloc_verbose =
    [4x3 double]    [8x4 double]    [4x3 double]
```

gmdp_see_policy_graph

Description

Determines and visualizes the actions prescribed by a policy for a given state of the global system.

Syntax

```
actions = gmdp_see_policy_graph(GMDP, policyloc, states)
actions = gmdp_see_policy_graph(GMDP, policyloc, states, iscoloraction)
```

Arguments

- **GMDP** : structure (see paragraph 2.1) defining the problem
- **policyloc** : a policy (see paragraph 2.3 for a description of the structure)
- **states** : vector (1xnb_site) of site states.
- **iscoloraction** : if true: node color related to action and state written in node, if false: node color related to state, action written in node (default value true)

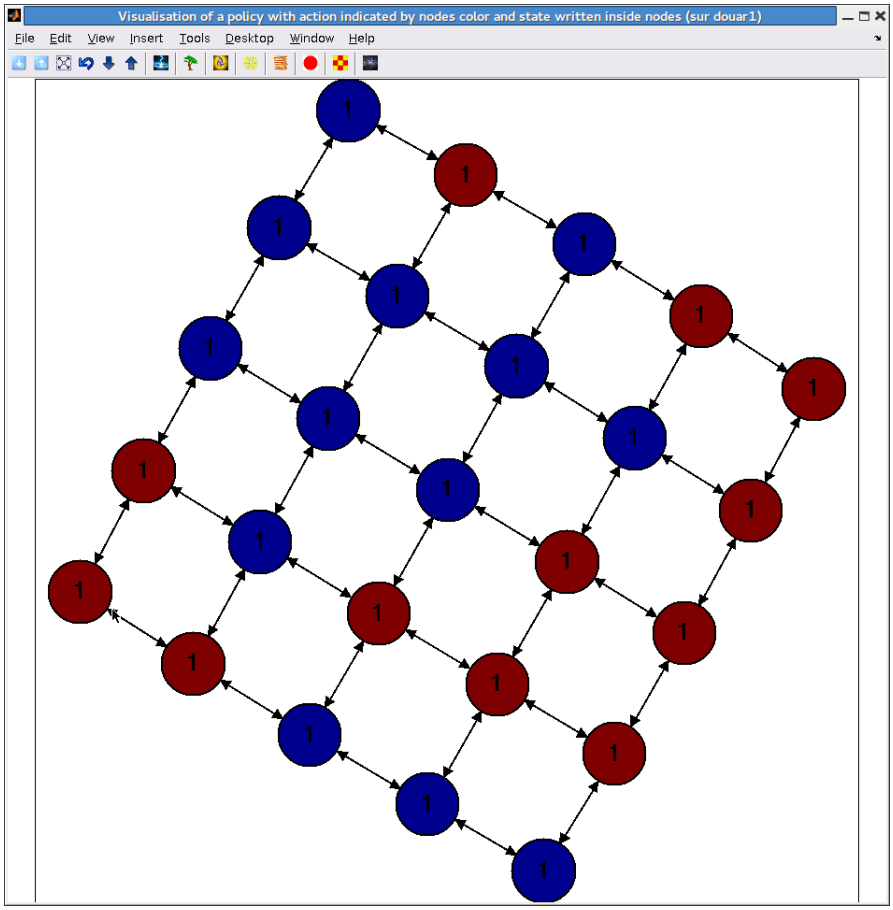
Evaluation

- **actions** : vector (1xnb_site) of site actions prescribed by policyloc.
- Displays information in a figure.

Example

```
>> rng(1); GMDP = gmdp_example_grid(5);
>> policyloc = gmdp_linear_programming(GMDP,0.9);
>> actions = gmdp_see_policy_graph(GMDP, policyloc, ones(1,25))
actions =
  Columns 1 through 15
     1   1   1   2   2   2   1   1   1   2   1   1   1   2   1
  Columns 16 through 25
     2   1   2   2   1   2   2   2   2   1
```

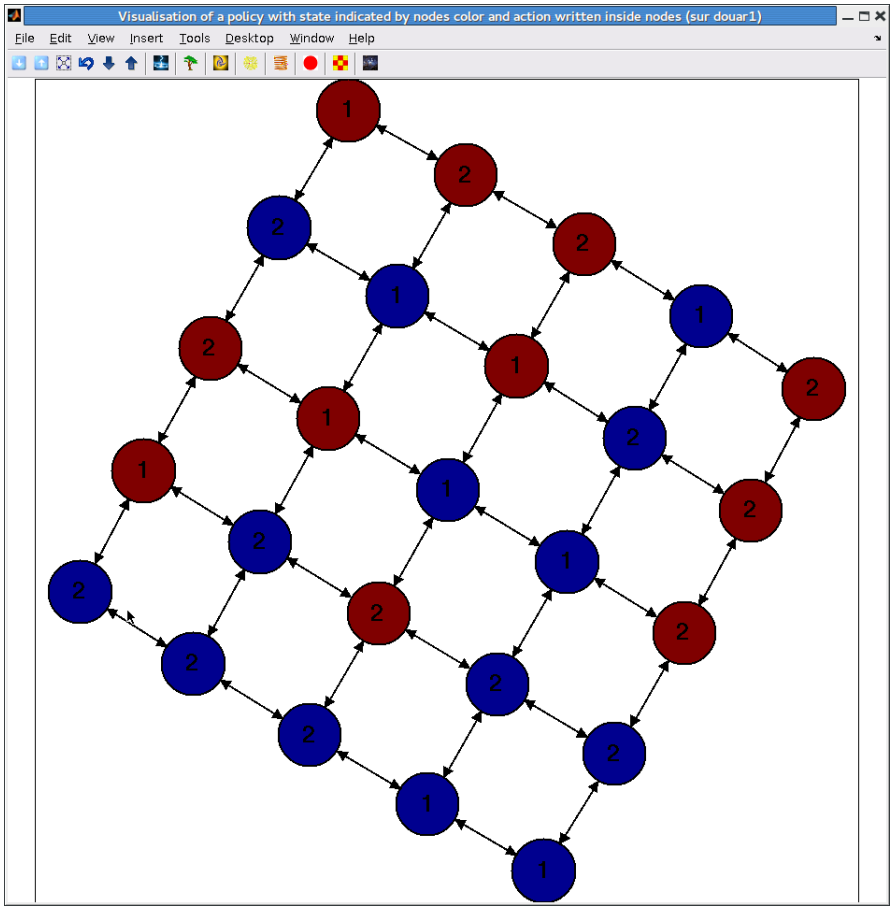
The following figure is also displayed.



We can see that when all sites are in state 1, policyloc prescribes action 1 in two clusters of sites and action 2 in the other sites.
In the second example below, the system state is set randomly and node color encodes site state while action prescribed is written inside the node.

```
>> actions = gmdp_see_policy_graph(GMDP, policyloc, randi(2,1,25), false)
actions =
  Columns 1 through 15
     1   2   2   1   2   2   1   1   2   2   2   1   1   2   2
  Columns 16 through 25
     1   2   1   2   1   2   2   2   2   1
```

The following figure is also displayed.



note that when right clicking on a node, it is colored in light yellow. This functionality may be usefull to enlight particular sites.

gmdp_analyze_policy

Description

Computes and displays actions repartition for a given policy globally for all sites and also for each site of a given subset of all sites. For a particular site, repartition is computed over all possible neighborhood states and all possible site states.

Syntax

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

Arguments

- **GMDP** : structure (see paragraph 2.1) defining the problem
- **policyloc** : a policy (see paragraph 2.3 for a description of the structure)
- **sites** : vector of sites to be considered (default value: all sites)

Evaluation

- **actions_repartition** : cell array (1x(n+1)) that contains proportions of actions for each site and for all sites considering the given policy. If a site has nit be considered, the corresponding

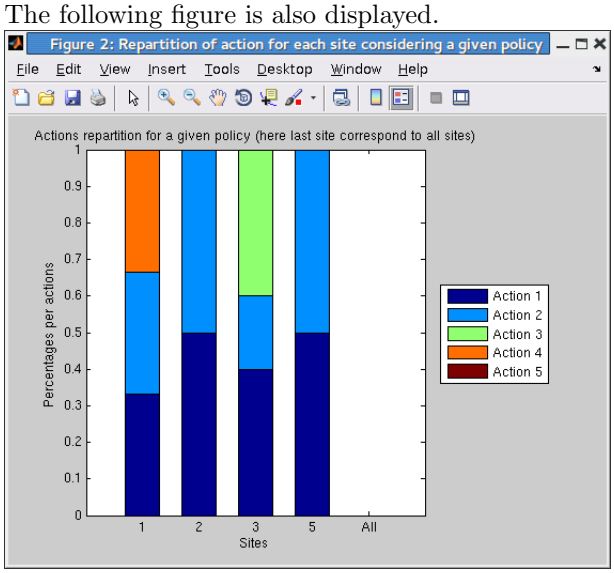
cell is empty. If sites have not the same number of possible actions, the last cell (corresponding to all sites) is empty.

- Displays information in a figure.

Example

```
>> rng(1); GMDP = gmdp_example_rand(5,[2 3 4 2 5], [5 2 4 3 2]);
>> policyloc = gmdp_linear_programming(GMDP,0.9);
>> actions_repartition = gmdp_analyze_policy(GMDP, policyloc,[1 2 3 5])
actions_repartition =
   [1x5 double]   [1x2 double]   [1x4 double]   []   [1x2 double]   []
```

The following figure is also displayed.



Here the global (on all sites) repartition of actions prescribed by the policy (the 'All' bar on the figure) cannot be computed because all sites do not have the same number of possible actions.

gmdp_analyze_policy_neighbor

Description

Computes and shows actions repartition for each state of each site considering a given policy.

Syntax

```
actions_repartition_state_site = gmdp_analyze_policy_neighbor(GMDP, policyloc)
actions_repartition_state_site = gmdp_analyze_policy_neighbor(GMDP, policyloc, sites)
```

Arguments

- **GMDP** : structure (see paragraph 2.1) defining the problem
- **policyloc** : a policy (see paragraph 2.3 for a description of the structure)
- **sites** : vector of sites to be considered (default value: all sites)

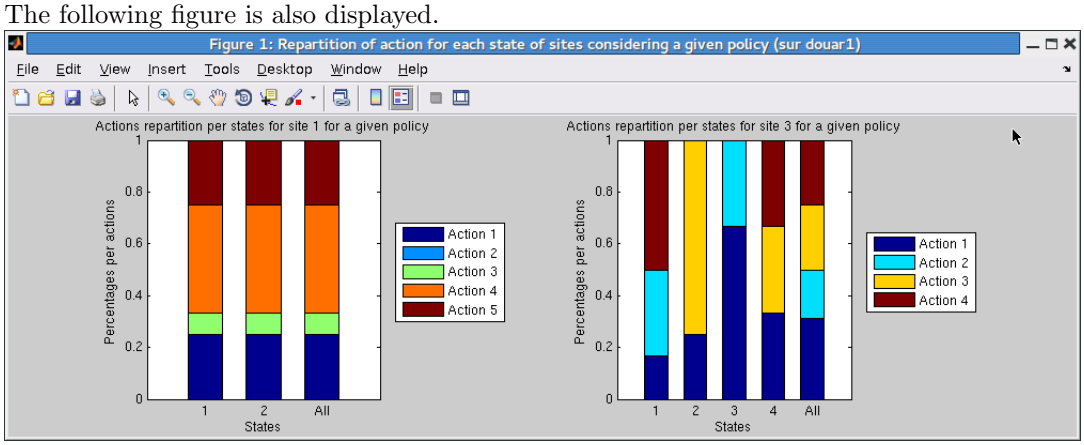
Evaluation

- **actions_repartition_state_site** : cell array (1xn) that contains proportions of actions for each state of each site and for all sites considering the given policy.
- Displays information in a figure limited to the 12 first sites.

Example

```
>> rng(2); GMDP = gmdp_example_rand(5,[2 3 4 2 5], [5 2 4 3 2]);
>> policyloc = gmdp_linear_programming(GMDP,0.9);
>> actions_repartition_state_site = gmdp_analyze_policy_neighbor(GMDP,policyloc,[1 3])
actions_repartition_state_site =
   [3x5 double]    []    [5x4 double]    []    []
```

The following figure is also displayed.



Here, for each considered sites, a plot shows the repartition of actions prescribed by the policy. A 'All' bar on the figure, is for all states of a site.

Policy evaluation functions

Mean-field based evaluation of a policy global value function

gmdp_eval_policy_MF

Description

Computes an approximation of the global value function of a policy, using the mean-field approximation of the transition probabilities.

Syntax

```
Vloc = gmdp_eval_policy_MF (GMDP, discount, policyloc)
Vloc = gmdp_eval_policy_MF (GMDP, discount, policyloc, Qloc0)
Vloc = gmdp_eval_policy_MF (GMDP, discount, policyloc, Qloc0, Tend)
```

Arguments

- **GMDP** : structure (see paragraph 2.1) defining the problem
- **discount** : discount rate in]0; 1[
- **policyloc** : a policy (see paragraph 2.3 for a description of the structure)
- **Qloc0** : proposal probability distributions for sites states at $t = 0$ (default value uniform for each site) Qloc0 is a cell array (1xn) of site state probability which is stored for site i in a vector of length $|\mathcal{S}_i|$.
- **Tend** : approximation of infinity (default value: 100)

Evaluation

- **Vloc** : set of local value functions (see paragraph 2.3 for a description of the structure)

Example

```
>> GMDP=gmdp_example_epidemio();
>> policyloc = gmdp_linear_programming (GMDP, 0.9);
>> Vloc = gmdp_eval_policy_MF (GMDP, 0.9, policyloc)
Vloc =
    [4x1 double]    [8x1 double]    [4x1 double]
```

Simulation-based evaluation of a policy global value function

gmdp_simulate_policy

Description

Simulates the system evolution when applying a given policy

Syntax

```
[sim_state, sim_reward, sim_action] = gmdp_simulate_policy(GMDP, policyloc)
[sim_state, sim_reward, sim_action] = gmdp_simulate_policy(GMDP, policyloc, nb_init)
[sim_state, sim_reward, sim_action] = gmdp_simulate_policy(GMDP, policyloc, nb_init, nb_run)
[sim_state, sim_reward, sim_action] = gmdp_simulate_policy(GMDP, policyloc, nb_init, nb_run, nb_peri
[sim_state, sim_reward, sim_action] = gmdp_simulate_policy(GMDP, policyloc, nb_init, nb_run, nb_peri
```

Arguments

- **GMDP** : structure (see paragraph 2.1) defining the problem
- **policyloc** : a policy (see paragraph 2.3 for a description of the structure)
- **nb_init** : number of initial states to consider (default value: 100)
- **nb_run** : number of runs for an initial state (default value: 100)
- **nb_period** : simulation length, i.e. number of periods (time steps) in a simulation (default value: 40)
- **state_init** : initial state, taken into account if nb_init=1 (default value: random)

Evaluation

- **sim_state** : array (nb_simulation x nb_period x nb_site) of simulated states for each simulation, each period and each site.
- **sim_reward** : array (nb_simulation x nb_period x nb_site) of simulated reward for each simulation, each period and each site.
- **sim_action** : array (nb_simulation x nb_period x nb_site) of simulated action for each simulation, each period and each site.

Example

```
>> GMDP = gmdp_example_epidemio();
>> policyloc = gmdp_linear_programming(GMDP,0.9);
>> [sim_state, sim_reward] = gmdp_simulate_policy(GMDP, policyloc,10,10);
>> size(sim_state)
ans =
   100    40     3
>> size(sim_reward)
ans =
   100    40     3
```

gmdp_eval_policy_value

Description

Computes and displays the mean global value evolution over time from simulations of the system evolution when applying a given policy.

Syntax

```
value_evolution = gmdp_eval_policy_value(discount, sim_reward)
value_evolution = gmdp_eval_policy_value(discount, sim_reward, iscumulated)
```

Arguments

- **discount** : discount rate in]0; 1[, or 1 when no discount is applied
- **sim_reward** : array (nb_simulation x nb_period x nb_site) of simulated evolution reward
- **iscumulated** : if true: computes cumulative discounted global value, if false: compute instantaneous global value (default value: true)

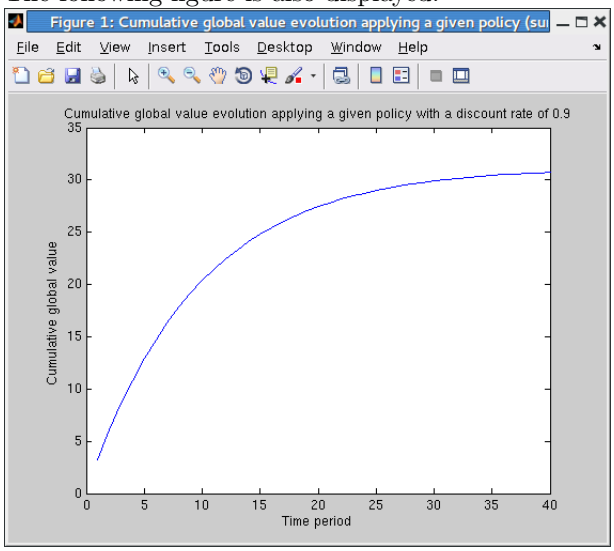
Evaluation

- **value_evolution** : vector (1xnb_period) of mean global (i.e. summed on all sites) value at each period
- Displays information on a figure.

Example

```
>> rng(4); GMDP = gmdp_example_rand(5,[2 3 4 2 5], [5 2 4 3 2]);
>> policyloc = gmdp_linear_programming(GMDP,0.9);
>> [sim_state, sim_reward] = gmdp_simulate_policy(GMDP, policyloc,10,10,40);
>> value_evolution = gmdp_eval_policy_value(0.9, sim_reward)
value_evolution =
 Columns 1 through 9
  3.2251   6.0744   8.6253  10.8441  12.9019  14.7035  16.3607  17.8437  19.1627
 Columns 10 through 18
  20.3850  21.4489  22.4310  23.3092  24.1109  24.8326  25.4716  26.0499  26.5605
 Columns 19 through 27
  27.0271  27.4497  27.8190  28.1583  28.4624  28.7361  28.9852  29.2037  29.4085
 Columns 28 through 36
  29.5873  29.7487  29.8927  30.0230  30.1410  30.2479  30.3444  30.4300  30.5081
 Columns 37 through 40
  30.5781  30.6410  30.6972  30.7493
```

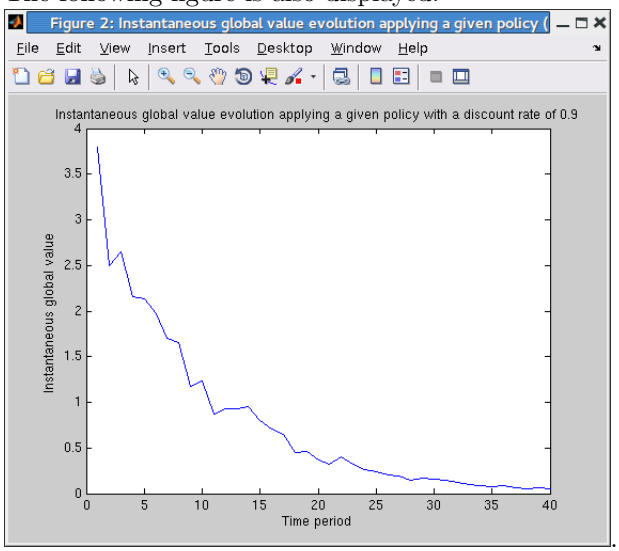
The following figure is also displayed.



The graph is smooth here, because a number of simulations was sufficiently large for a good estimation. The curve reaches a plateau, which means that the number of periods is large enough for a good infinite horizon value estimation.
The function can also be used with a single simulation. It enables to display instantaneous reward evolution:

```
>> value_evolution = gmdp_eval_policy_value(0.9, sim_reward(1,:,:), false)
value_evolution =
  Columns 1 through 9
  3.8003  2.4983  2.6477  2.1593  2.1319  1.9738  1.6977  1.6506  1.1789
  Columns 10 through 18
  1.2315  0.8692  0.9295  0.9267  0.9513  0.8024  0.7133  0.6480  0.4503
  Columns 19 through 27
  0.4640  0.3703  0.3176  0.4081  0.3253  0.2636  0.2404  0.2061  0.1964
  Columns 28 through 36
  0.1437  0.1699  0.1636  0.1418  0.1272  0.0957  0.0876  0.0799  0.0841
  Columns 37 through 40
  0.0589  0.0517  0.0698  0.0502
```

The following figure is also displayed.



We can observe that the discounted global reward is almost 0 after 40 periods for a discount of 0.9. For a larger discount, more periods should have been considered.

gmdp_eval_policy_value_site_contribution

Description

Estimates the contributions of each to the cumulative (over time) global value from simulation of the system evolution when applying a given policy.

Syntax

```
value_site = gmdp_eval_policy_value_site_contribution(GMDP, discount, sim_reward)
```

Arguments

- **GMDP** : structure (see paragraph 2.1) defining the problem
- **discount** : discount rate in]0; 1[, or 1 when no discount is applied
- **sim_reward** : array (nb_simulation x nb_period x nb_site) of simulated reward for each simulation, each period and each site

Evaluation

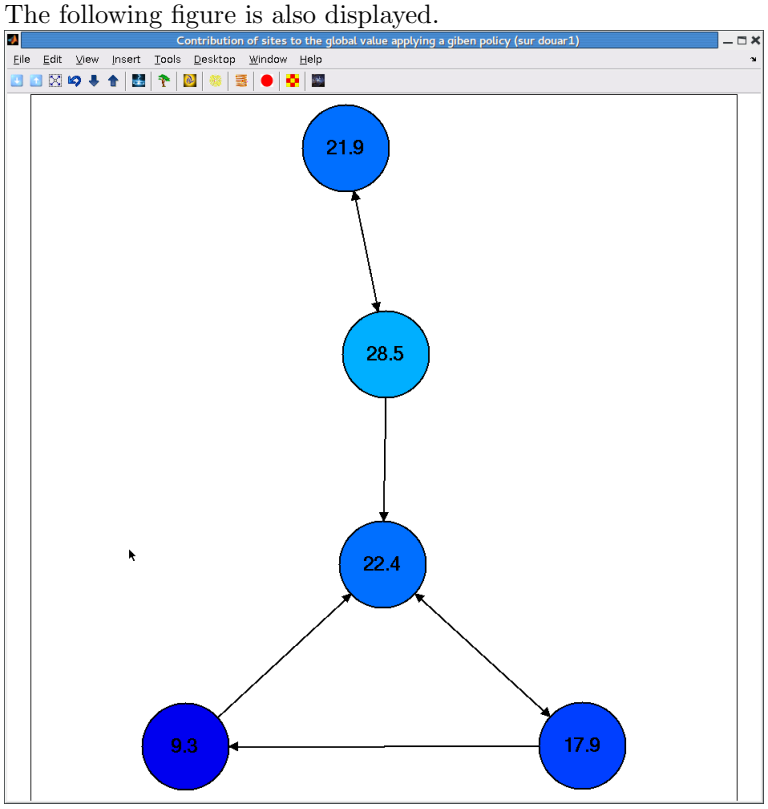
- **value_site** : vector (1xnb_site) of contribution of each site to the (discounted or not) cumulative global value

- Displays information on a figure.

Example

```
>> rng(5); GMDP = gmdp_example_rand(5,[2 3 4 2 5], [5 2 4 3 2]);
>> policyloc = gmdp_linear_programming(GMDP,0.9);
>> [sim_state, sim_reward] = gmdp_simulate_policy(GMDP, policyloc,10,10,40);
>> value_site = gmdp_eval_policy_value_site_contribution(GMDP, 0.9, sim_reward)
value_site =
   28.4900   17.9097   22.4354   21.8572    9.3076
```

The following figure is also displayed.



Note that by double-clicking a particular sit, the site number is displayed.

gmdp_eval_policy_state_time

Description

Computes and displays the time spent in each site's state from simulations of the system when applying a given policy.

Syntax

```
state_time = gmdp_eval_policy_state_time(GMDP, sim_state)
state_time = gmdp_eval_policy_state_time(GMDP, sim_state, sites)
```

Arguments

- **GMDP** : structure (see paragraph 2.1) defining the problem
- **sim_state** : array (nb_simultion x nb_period x nb_site) of simulated evolution states
- **sites** : vector of sites to be considered (default value: all sites)

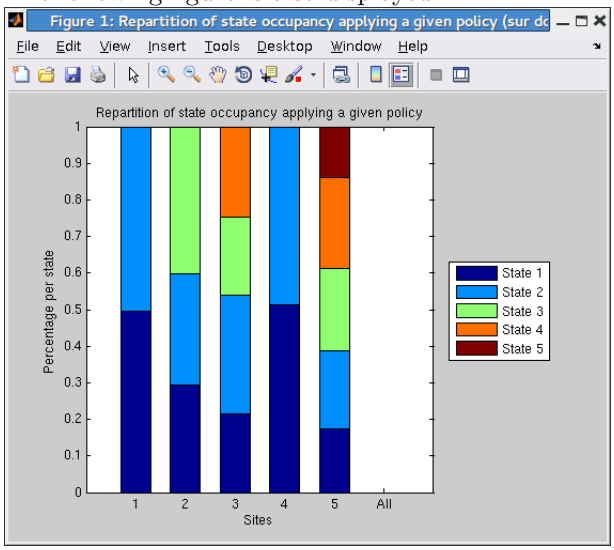
Evaluation

- **state_time** : cell array (1x(n+1)) that contains vector of percentage of time spend in each possible state of sites. If a site is not considered the vector is empty The last cell element called 'All' is computed only if all sites have the same number of states and if argument sites defined all sites.
- Displays information in a figure.

Example

```
>> rng(3); GMDP = gmdp_example_rand(5,[2 3 4 2 5], [5 2 4 3 2]);
>> policyloc = gmdp_linear_programming(GMDP,0.9);
>> [sim_state, sim_reward] = gmdp_simulate_policy(GMDP, policyloc,10,10,40);
>> state_time = gmdp_eval_policy_state_time(GMDP, sim_state)
state_time =
   [2x1 double]    [3x1 double]    [4x1 double]    [2x1 double]    [5x1 double]    []
```

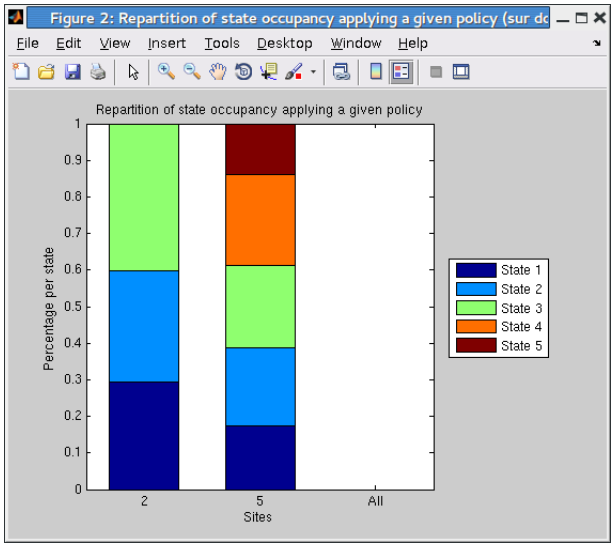
The following figure is also displayed.



With this second example e now plot only informations for sites 2 and 5.

```
>> state_time = gmdp_eval_policy_state_time(GMDP, sim_state, [2 5])
state_time =
    []    [3x1 double]    []    []    [5x1 double]    []
```

The following figure is also displayed.



Evaluation of global value function

gmdp_eval_policy_global_value

Description

Computes the global value function for all possible initial state of the system, associated to a set of local value functions. Only possible when the global state space is not too large!

Syntax

```
V = gmdp_eval_policy_global_value (GMDP, Vloc)
```

Argument

- **GMDP** : structure (see paragraph 2.1) defining the problem
- **Vloc** : set of local value functions (see paragraph 2.3 for a description of the structure)

Evaluation

- **V** : global value function, vector of size equal to $|S|$

Example

```
>> GMDP = gmdp_example_epidemio(); discount = 0.9;
>> policyloc = gmdp_linear_programming (GMDP, discount);
>> Vloc = gmdp_eval_policy_MF (GMDP, discount, policyloc);
Vloc =
    [4x1 double]    [8x1 double]    [4x1 double]
>> V = gmdp_eval_policy_global_value (GMDP, Vloc)
V =
   84.9081
   81.5414
   81.9591
   78.5924
   81.5414
   78.1747
   78.5924
   75.2257
```

gmdp_eval_policy_global_value_state

Description

Computes the global value for a particular initial state of the system, associated to a set of local value functions.

Syntax

```
v = gmdp_eval_policy_global_value_state (GMDP, Vloc, v_s)
```

Argument

- **GMDP** : structure (see paragraph 2.1) defining the problem
- **Vloc** : approximate local value function (see paragraph 2.3 for a description of the structure)
- **v_s** : a global state given as a vector (1xn) of sites states

Evaluation

- **v** : global value

Example

```
>> GMDP = gmdp_example_epidemio(); discount = 0.9;
>> policyloc = gmdp_linear_programming (GMDP, discount);
>> Vloc = gmdp_eval_policy_MF (GMDP, discount, policyloc);
Vloc =
    [4x1 double]    [8x1 double]    [4x1 double]
>> v = gmdp_eval_policy_global_value_state (GMDP, Vloc, [2 1 2])
v =
   78.1747
```

Utilities

gmdp_check_gmdp

Description

gmdp_check_gmdp checks whether the GMDP defined by the GMDP structure is valid. If the GMDP provided structure is not valid, then the function returns a list of error messages describing the problems. This function also pre-compute the two attributes N and VN of the structure GMDP if not yet available (see paragraph 2.2).

Syntax

```
[isOK, problems, GMDP] = gmdp_check_gmdp(GMDP)
```

Argument

- **GMDP** : structure (see paragraph 2.1) defining a problem

Evaluation

- **is_OK** : `true` if GMDP is valid, `false` otherwise
- **problems** : cell array of error message(s)

Example

```
>> % Generate a GMDP and check validity
>> GMDP = gmdp_example_epidemio();
>> gmdp_check_gmdp_gmdp(GMDP)
ans =
     1
problems =
    {[]}

>> % Introduce error and check validity
>> GMDP.G(1,1)=2;
>> [isOK, problems] = gmdp_check_gmdp(GMDP)
isOK =
     0
problems =
    'GMDP Toolbox ERROR: G (adjacency matrix nxn of sites neighborhood,
     with n the number of sites) must contain only 0 and 1'
```

gmdp_check_policy

Description

Checks to validity of a GMDP problem and a policy for this problem.

Syntax

```
[isOK, problems] = gmdp_check_policy(GMDP, policyloc, ischecked)
```

Argument

- **GMDP** : structure (see paragraph 2.1) defining a problem
- **policyloc** : a policy (see paragraph 2.3 for a description of the structure)
- **ischecked** : if true, check GMDP ; otherwise, do not check GMDP (default value: true)

Evaluation

- **is_OK** : `true` if GMDP is valid, `false` otherwise
- **problems** : cell array of error message(s)

Example

```
>> % Generate a GMDP and check validity
>> GMDP = gmdp_example_epidemio();
>> policyloc = gmdp_linear_programming(GMDP,0.9);
>> gmdp_check_policy(GMDP, policyloc)
ans =
     1
problems =
    {[]}

>> % Introduce error and check validity
>> policyloc{1}(3)=5;
>> [isOK, problems] = gmdp_check_policy(GMDPpolicyloc, false)
isOK =
     0
problems =
   'GMDP Toolbox ERROR: policyloc is not valid for site(s) 1'
```

gmdp_globalize_local_policy

Description

Converts a local policy in global policy.

Syntax

```
policy = gmdp_globalize_local_policy(GMDP,policyloc)
```

Arguments

- **GMDP** : structure (see paragraph 2.1) defining the problem
- **policyloc** : a policy (see paragraph 2.3 for a description of the structure)

Evaluation

- **policy** : a global policy: a vector (($prod_{k\in 1...n}|\mathcal{S}_k|$)x1).

Example

```
>> rng(0); GMDP= gmdp_example_rand(3,[3 4 5], [5 4 3]);
>> policyloc = gmdp_linear_programming (GMDP, 0.95);
>> policy = gmdp_globalize_local_policy(GMDP,policyloc)
policy =
     1
     2
     3
     4
     5
     6
     7
     8
```

gmdp_localize_global_policy

Description

If possible, converts a global policy in local policy.

Syntax

```
[islocal policyloc] = gmdp_localize_global_policy(GMDP, policy)
```

Arguments

- **GMDP** : structure (see paragraph 2.1) defining the problem
- **policy** : a global policy: a vector (($prod_{k\in 1...n}|\mathcal{S}_k|$)x1).

Evaluation

- **is_local** : boolean indicating if a local policy exists
- **policyloc** : a policy (see paragraph 2.3 for a description of the structure

Example

```
>> rng(0); GMDP= gmdp_example_rand(3,[3 4 5], [5 4 3]);
>> policyloc = gmdp_linear_programming (GMDP, 0.95);
>> policy = gmdp_globalize_local_policy(GMDP,policyloc);
>> [islocal policyloc] = gmdp_localize_global_policy(GMDP, policy)
islocal =
     1
policyloc =
    [4x1 double]    [8x1 double]    [4x1 double]
```

gmdp_see_neighborhood

Description

Visualizes the neighborhood relations between sites.

Syntax

```
gmdp_see_neighborhood(GMDP)
gmdp_see_neighborhood(GMDP, sites)
```

Arguments

- **GMDP** : structure (see paragraph 2.1) defining the problem
- **sites** : vector of sites that will be displayed with a color different to the default color (default value: empty)

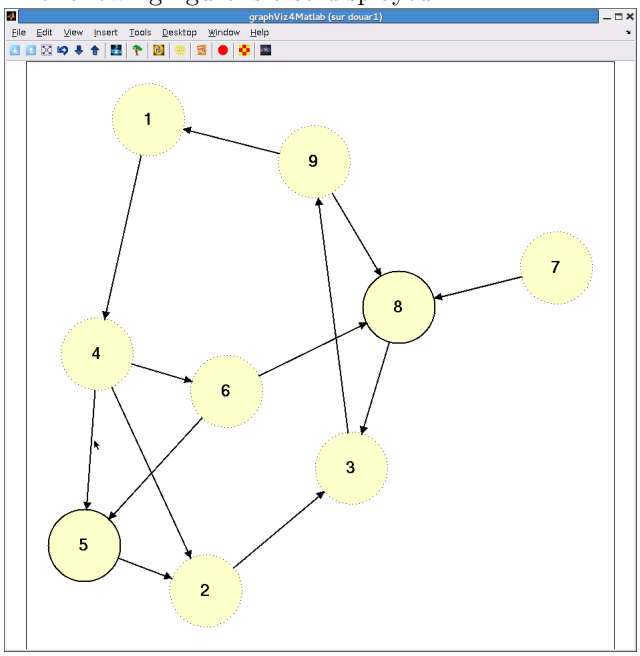
Evaluation

- Displays information in a figure. Note that a black circle surrounding a node represents a self loop (i.e. $i \in N_i$). In the examples below, only single-direction edges are present. See next section for an example with two-directions edges.

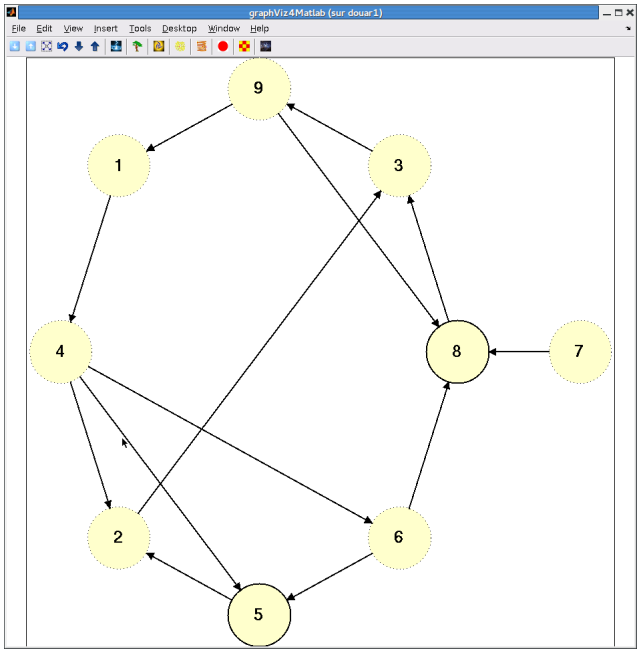
Example

```
>> rng(1); GMDP = gmdp_example_rand(10);
>> gmdp_see_neighborhood(GMDP)
```

The following figure is also displayed.



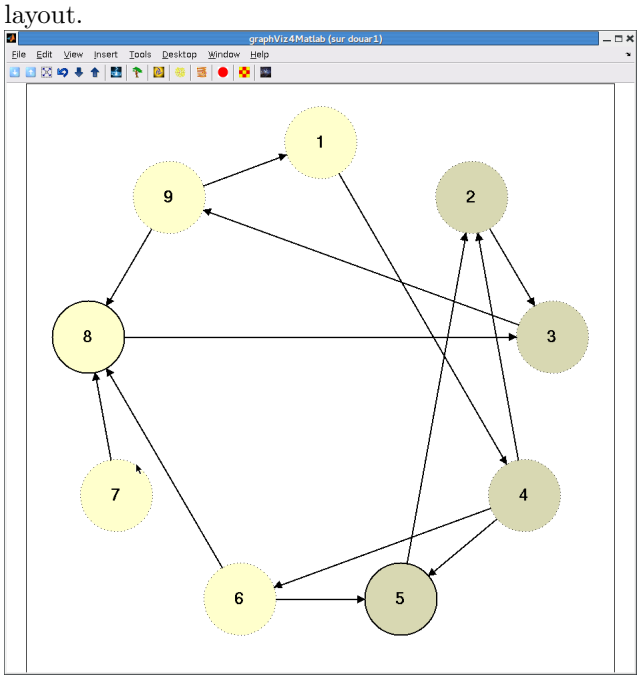
The layout can be changed in an interactive way. For instance a circular layout leads to he following figure.



It is possible to distinguish a set of site coloring them in another color. It can be useful for large size problems

```
>> gmdp_see_neighborhood(GMDP, 2:5)
```

The sites 2, 3, 4, 5 are now colored with a different color. Here is the view with the simple circular layout.



Bibliography

[1] M.L. Puterman, *Markov Decision Processses*. John Wiley and Sons, 1994.

[2] O. Sigaud, O. Buffet, *Processus Décisionnels de Markov en Intelligence Artificielle*. Germes, 2008.

[3] C. Boutilier, R. Dearden, M. Goldszmidt, *Stochastic dynamic programming with factored representations*. Artificial Intelligence 121(1), p 49-107, 2000.

[4] C. Guestrin, D. Koller, R. Parr, S. Venkataraman, *Efficient solution algorithms for factored MDPs*. Journal of Artificial Intelligence Research (19), p 399–468, 2003

[5] N. Forsell, R. Sabbadin, *Approximate linear-programming algorithms for graph-based Markov decision processes*. Proceedings of ECAI, p 590-594, 2006.

[6] N. Forsell, N. Peyrard, R. Sabbadin, *A framework and a Mean-Field algorithm for the local control of spatial processes*. International Journal of Approximate Reasoning, p 66-86, 2012.

[7] GMDPtoolbox site:
`http://inra.fr/mia/T/GMDPtoolbox`

[8] GMDPtoolbox code:
`https://mulcyber.toulouse.inra.fr/frs/?group_id=213`.

[9] Web site of the MDPtoolbox :
`http://inra.fr/mia/T/MDPtoolbox`.

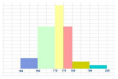
GMDPtoolbox functions

GMDP generation

gmdp_example_epidemio	Generates a tiny GMDP epidemics management problem on 3 crop fields
gmdp_example_grid	Generates a random GMDP problem on a grid of sites
gmdp_example_rand	Generates a random GMDP problem on a random network of sites

GMDP resolution

gmdp_linear_programming	Solves a GMDP using an approximate linear programming algorithm
gmdp_policy_iteration_MF	Solves a GMDP using the Mean-Field policy iteration algorithm
gmdp_globalize	Translates a GMDP problem into a MDP problem



Policy description

gmdp_see_policy	Displays actions prescribed by a policy
gmdp_see_policy_tab	Displays the tables of actions prescribed by a policy for all neighborhood states
gmdp_see_policy_graph	Determines and visualizes the actions prescribed by a policy for a given system state of the global system
gmdp_analyze_policy	Computes and displays actions repartition for each site for a given policy
gmdp_analyze_policy_neighbor	Computes and displays actions repartition for each state of each site for a given policy

Policy evaluation

Mean-field based evaluation

gmdp_eval_policy_MF	Evaluates a policy using the Mean-Field approximation

Simulation based evaluation

gmdp_simulate_policy	Simulate trajectories of the system when applying a given policy
gmdp_eval_policy_value	Estimates the global value evolution over time when applying a policy
gmdp_eval_policy_value_site_contribution	Estimates the contribution of sites to the global value when applying a policy
gmdp_eval_policy_state_time	Computes and displays the time spent in each state when applying a policy

Evaluation of global value

gmdp_eval_policy_global_value	Computes the global value function for all possible initial state of the system, associated to a set of local value functions
gmdp_eval_policy_global_value_state	Computes the global value for a particular initial state of the system, associated to a set of local value functions

Utilities

gmdp_check_gmdp	Checks the validity of a GMDP
gmdp_check_policy	Checks the validity of a policy for a GMDP
gmdp_globalize_local_policy	Computes the global policy from a local policy
gmdp_localize_global_policy	If it exists, computes the local policy from a global policy
gmdp_see_neighborhood	Visualizes the neighborhood relations between sites