École Temps Réel '09

Une méthode inverse pour les processus de décision markoviens

Étienne André

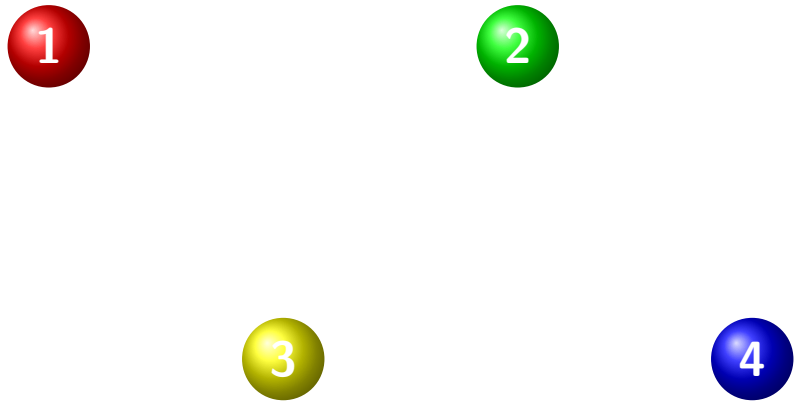
Laboratoire Spécification et Vérification
LSV, ENS de Cachan & CNRS

Contexte : vérification de systèmes temps réel

- Utilisation de méthodes formelles pour la vérification
- Vérification de systèmes temps réel à comportement stochastique
  - Besoin d'exprimer des probabilités
  - Besoin d'exprimer des comportement infinis
  - Utilisation des processus de décision markoviens [Bel57, How60]
- Besoin d'ajuster certaines valeurs temporelles du système
  - Utilisation de paramètres (constantes à valeur inconnue)
  - Définition de zones de bon fonctionnement pour les valeurs de ces paramètres

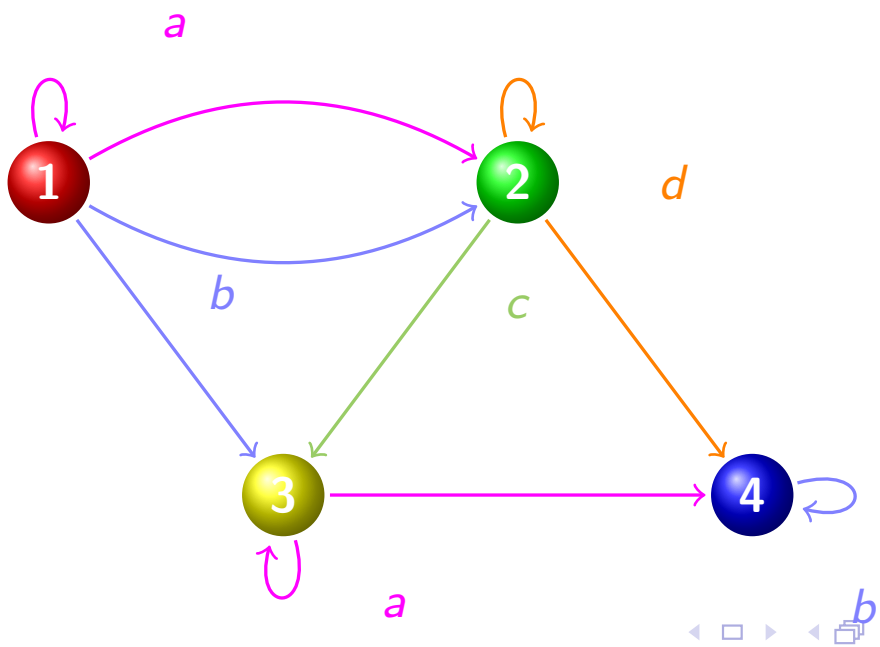
Processus de décision markovien (PDM)

- Graphe orienté pondéré auquel on ajoute des probabilités
  - Ensemble d'états $S = \{s_1, \ldots, s_n\}$



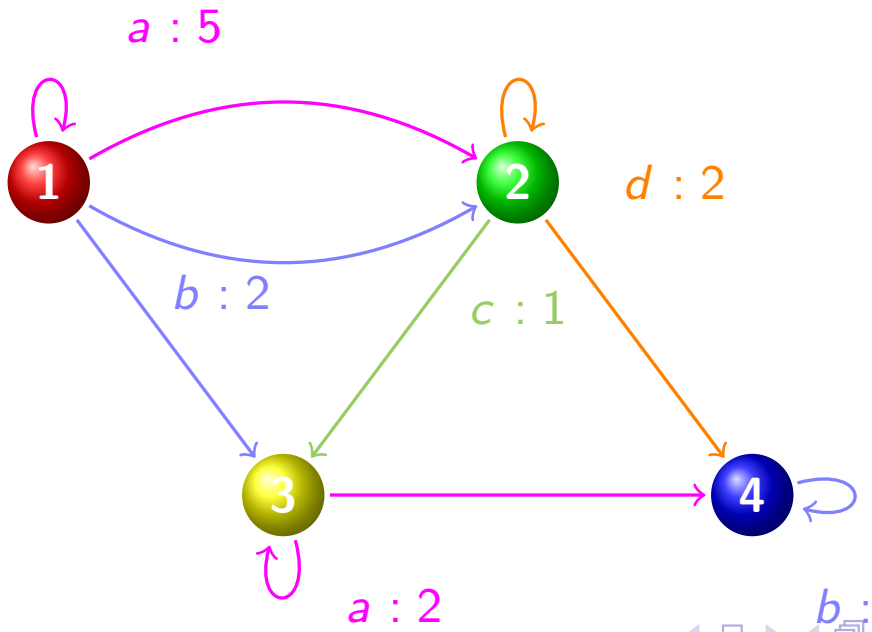
Processus de décision markovien (PDM)

- Graphe orienté pondéré auquel on ajoute des probabilités
  - Ensemble d'états $S = \{s_1, \ldots, s_n\}$
  - Ensemble A d'actions (ou étiquettes)



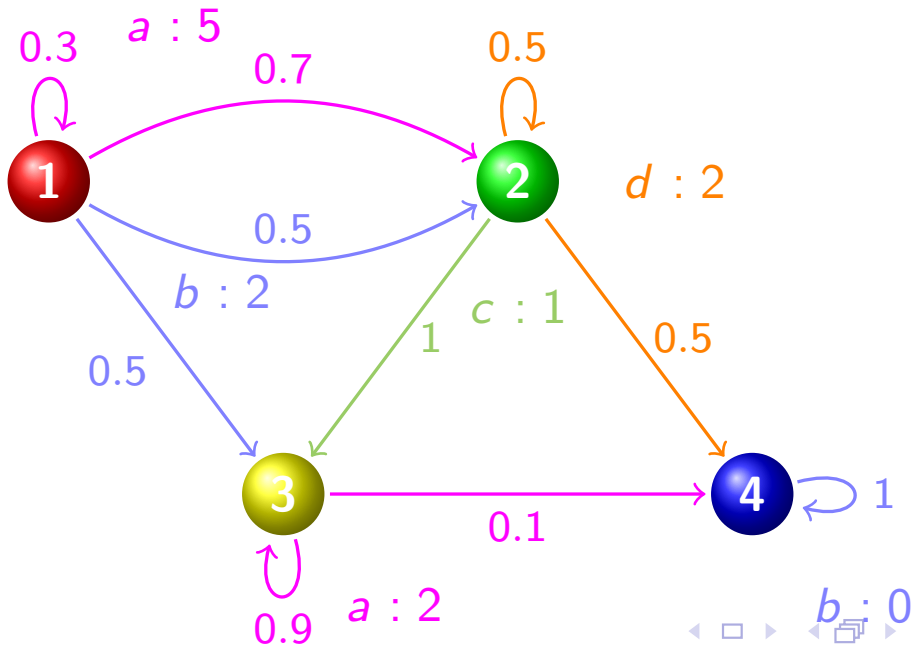
Processus de décision markovien (PDM)

- Graphe orienté pondéré auquel on ajoute des probabilités
  - Ensemble d'états $S = \{s_1, \ldots, s_n\}$
  - Ensemble A d'actions (ou étiquettes)
  - Fonction de coût w, associant un coût $w(s, a)$ à chaque état s et action a



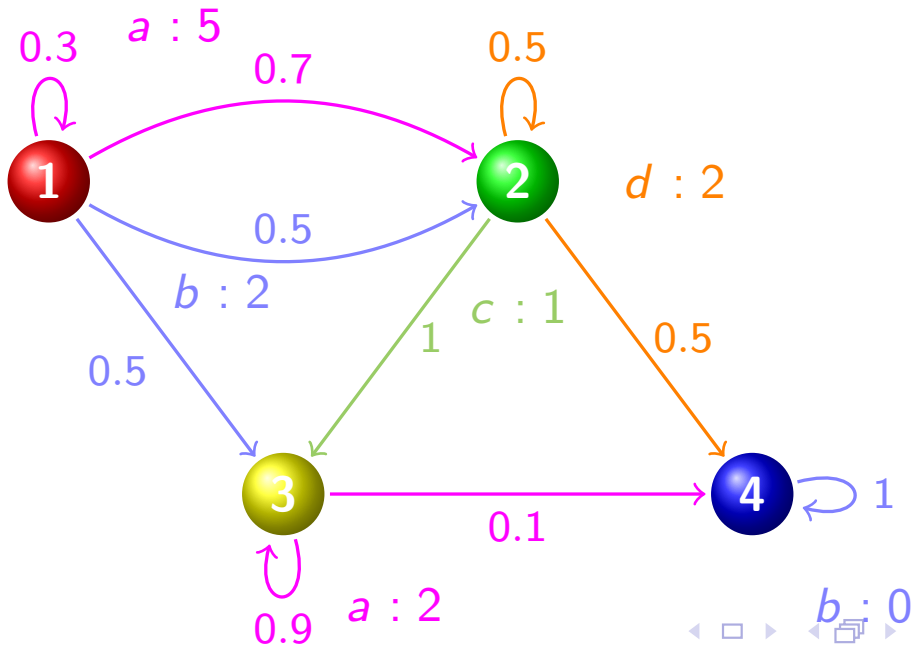
Processus de décision markovien (PDM)

- Graphe orienté pondéré auquel on ajoute des probabilités
  - Ensemble d'états $S = \{s_1, \ldots, s_n\}$
  - Ensemble A d'actions (ou étiquettes)
  - Fonction de coût w, associant un coût $w(s, a)$ à chaque état s et action a
  - Fonction de probabilité $Prob$, associant une probabilité à chaque arc, telle que la somme des probabilités quittant un état s via l'action a soit égale à 1, c'est-à-dire $\sum_{s' \in S} Prob(s, a, s') = 1$



Processus de décision markovien (PDM)

- Graphe orienté pondéré auquel on ajoute des probabilités
  - Ensemble d'états $S = \{s_1, \ldots, s_n\}$, comprenant un état puits s_n
  - Ensemble A d'actions (ou étiquettes)
  - Fonction de coût w, associant un coût $w(s, a)$ à chaque état s et action a
  - Fonction de probabilité $Prob$, associant une probabilité à chaque arc, telle que la somme des probabilités quittant un état s via l'action a soit égale à 1, c'est-à-dire $\sum_{s' \in S} Prob(s, a, s') = 1$

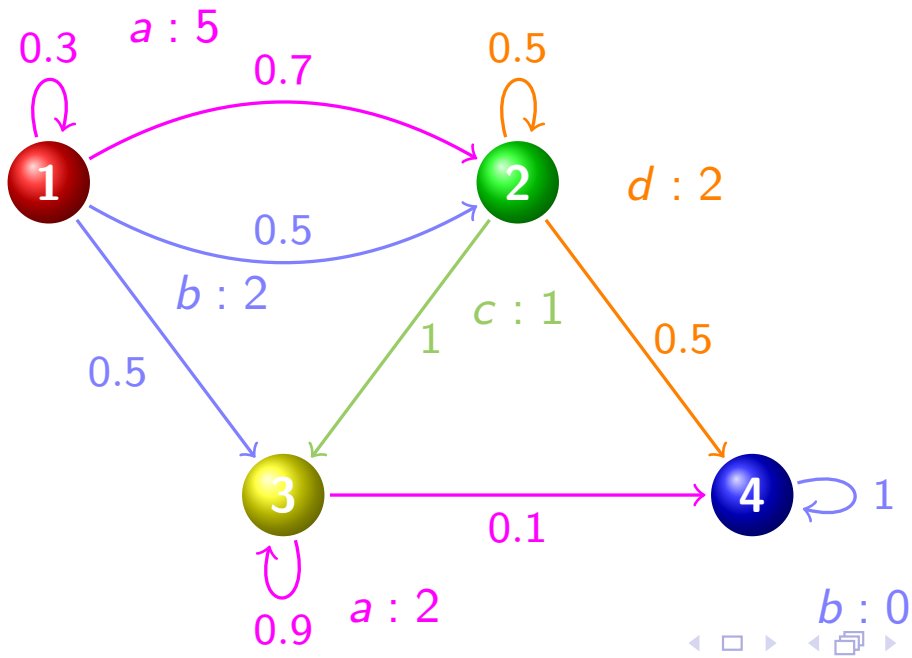


Le problème direct : politique optimale

- Politique μ : fonction des états aux actions $S \to A$
  - Supprime le non-déterminisme
  - Le PDM devient alors une chaîne de Markov [KMST59]
- Politique optimale : politique telle que la somme des coûts jusqu'à l'état puits est minimale

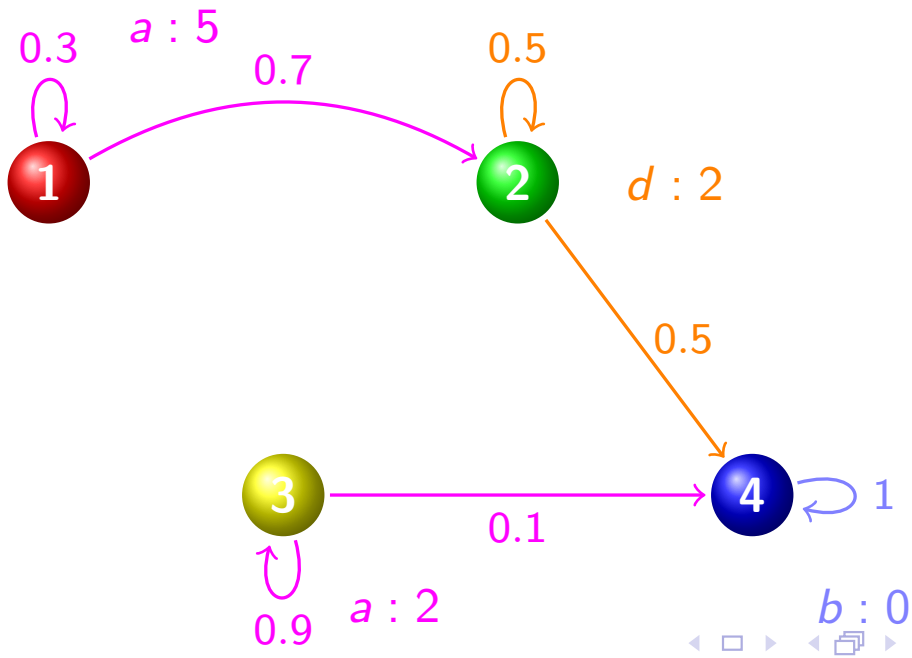
Le problème direct : politique optimale

- Politique μ : fonction des états aux actions $S \to A$
  - Supprime le non-déterminisme
  - Le PDM devient alors une chaîne de Markov [KMST59]
- Politique optimale : politique telle que la somme des coûts jusqu'à l'état puits est minimale
- Politique optimale pour notre exemple de PDM



Le problème direct : politique optimale

- Politique μ : fonction des états aux actions $S \to A$
  - Supprime le non-déterminisme
  - Le PDM devient alors une chaîne de Markov [KMST59]
- Politique optimale : politique telle que la somme des coûts jusqu'à l'état puits est minimale
- Politique optimale pour notre exemple de PDM
  - $\mu = \{1 \to a, \quad 2 \to d, \quad 3 \to a\}$



Le problème inverse

- Rappel : le problème direct
  - Étant donné un PDM, déterminer la politique optimale
- Le problème inverse
  - Étant donnés un PDM et une politique optimale, est-il possible de changer la valeur de certains des coûts, tout en conservant le fait que la politique optimale reste optimale ?

Le problème inverse

- Rappel : le problème direct
  - Étant donné un PDM, déterminer la politique optimale
- Le problème inverse
  - Étant donnés un PDM et une politique optimale, est-il possible de changer la valeur de certains des coûts, tout en conservant le fait que la politique optimale reste optimale ?
- Un peu plus formellement...

Objectif

Étant donnés un PDM $\mathcal{M}$ et une politique optimale μ_0, synthétiser une contrainte K_0 sur les coûts du système considérés comme des paramètres telle que, pour toute valuation des paramètres, la politique μ_0 reste optimale

Plan de l'exposé

Plan de l'exposé

L'algorithme classique de détermination de valeur

- Utilisé par l'algorithme d'itération de politiques pour déterminer la politique optimale
- Entrées
  - Un processus de décision markovien $\mathcal{M} = (S, A, Prob, w)$
  - Une politique μ
- Sortie
  - Une fonction de valeur v, associant une valeur numérique à chaque état s, c'est-à-dire le coût pour atteindre, depuis s, l'état puits dans le PDM $\mathcal{M}$ restreint à la politique μ

Algorithme (Détermination de valeur)

RÉSOUDRE $\{v(s) = w(s, \mu[s]) + \sum_{s' \in S} Prob(s, \mu[s], s') \times v(s')\}_{s \in S \setminus s_n}$

L'algorithme de détermination de valeur : application

Algorithme (Détermination de valeur)

RÉSOUDRE $\{v(s) = w(s, \mu[s]) + \sum_{s' \in S} Prob(s, \mu[s], s') \times v(s')\}_{s \in S \setminus s_n}$

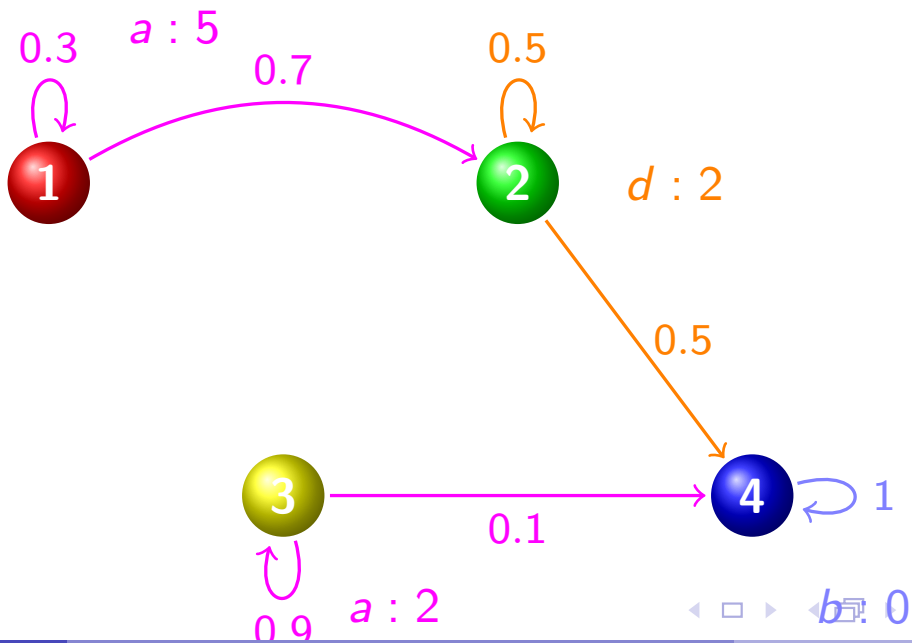
$\mu = \{1 \to a, \quad 2 \to d, \quad 3 \to a\}$

$v(1) = w(1, a) + 0.3 \times v(1) + 0.7 \times v(2)$

$v(2) = w(2, d) + 0.5 \times v(2) + 0.5 \times v(4)$

$v(3) = w(3, a) + 0.9 \times v(3) + 0.1 \times v(4)$

$v(4) = 0$



L'algorithme de détermination de valeur : application

Algorithme (Détermination de valeur)

RÉSOUDRE $\{v(s) = w(s, \mu[s]) + \sum_{s' \in S} Prob(s, \mu[s], s') \times v(s')\}_{s \in S \setminus s_n}$

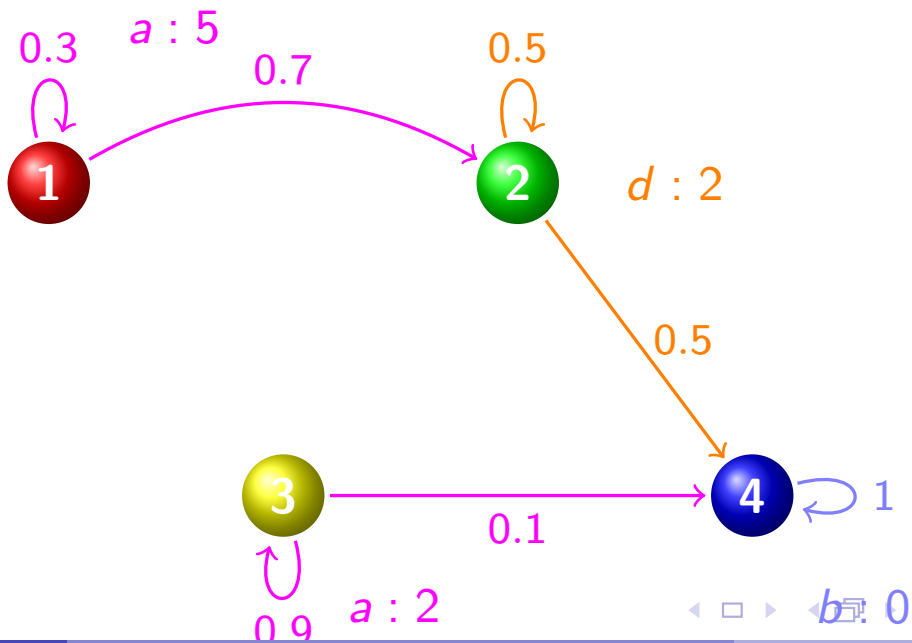
$\mu = \{1 \to a, \quad 2 \to d, \quad 3 \to a\}$

$v(1) = w(1, a) + 0.3 \times v(1) + 0.7 \times v(2)$

$v(2) = w(2, d) + 0.5 \times v(2) + 0.5 \times v(4) \quad = 4$

$v(3) = w(3, a) + 0.9 \times v(3) + 0.1 \times v(4) \quad = 20$

$v(4) = 0$



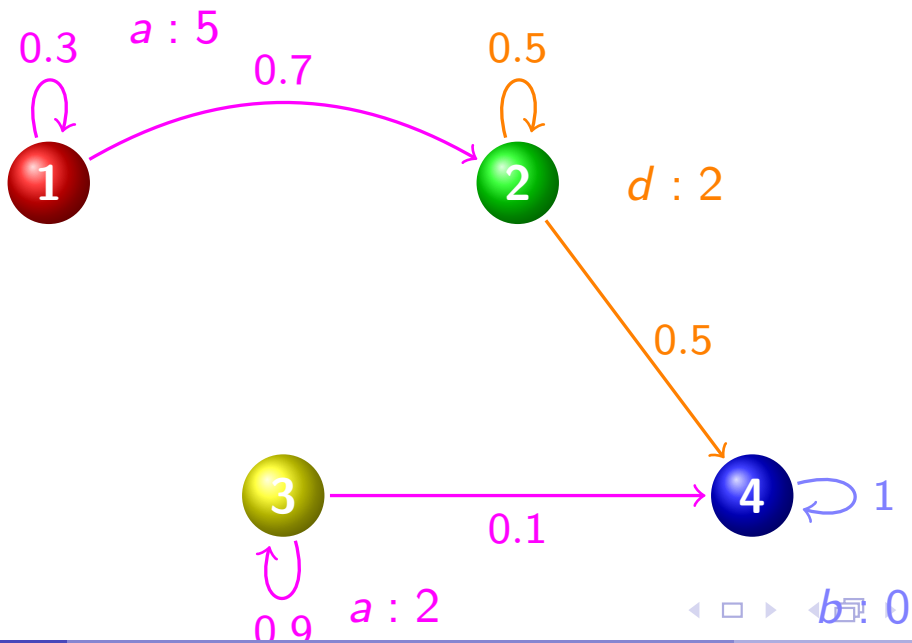
L'algorithme de détermination de valeur : application

Algorithme (Détermination de valeur)

RÉSOUDRE $\{v(s) = w(s, \mu[s]) + \sum_{s' \in S} Prob(s, \mu[s], s') \times v(s')\}_{s \in S \setminus s_n}$

$\mu = \{1 \to a, \quad 2 \to d, \quad 3 \to a\}$

$$
\begin{aligned}
v(1) &= w(1,a) + 0.3 \times v(1) + 0.7 \times v(2) &&= \frac{78}{7} \\
v(2) &= w(2,d) + 0.5 \times v(2) + 0.5 \times v(4) &&= 4 \\
v(3) &= w(3,a) + 0.9 \times v(3) + 0.1 \times v(4) &&= 20 \\
v(4) &= 0
\end{aligned}
$$



L'algorithme classique d'itération de politiques

- Entrée : un processus de décision markovien $\mathcal{M} = (S, A, Prob, w)$
- Sortie : une politique optimale μ
- Principe :
  1. Débuter avec une politique quelconque
  2. Calculer la fonction de valeur à l'aide de l'algorithme *ValueDet*
  3. Calculer une meilleure politique, et aller à (2) jusqu'à point fixe

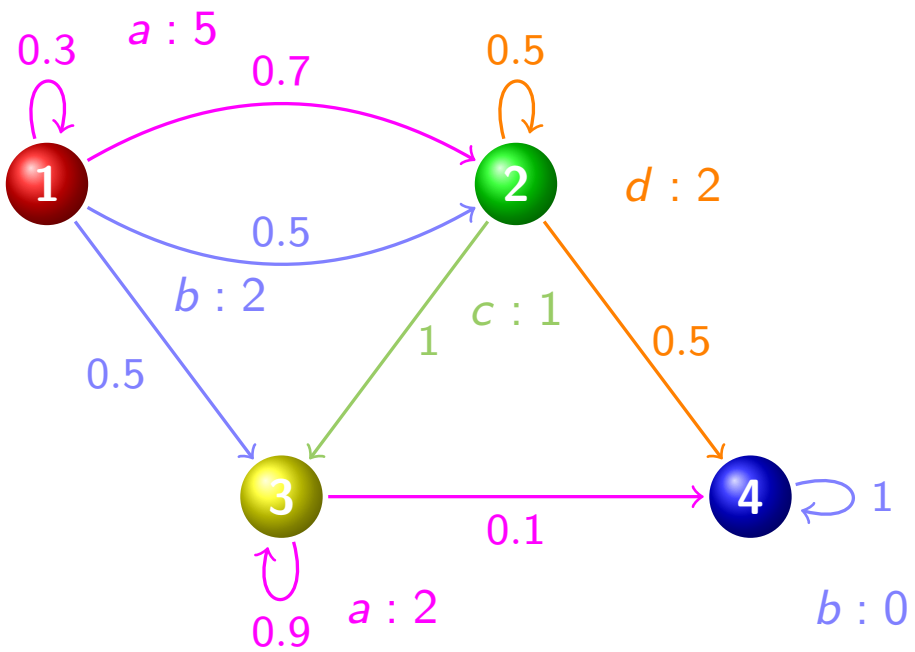
Algorithme (Itération de politiques)

RÉPÉTER JUSQU'À POINT FIXE
$v := ValueDet(M, \mu)$
pour tout $s \in S \setminus s_n$ **FAIRE**
$optimum := v[s]$
pour tout $a \in e(s)$ **FAIRE**
SI $w(s,a) + \sum_{s' \in S} Prob(s,a,s')v(s') < optimum$ **ALORS**
$optimum := w(s,a) + \sum_{s' \in S} Prob(s,a,s')v(s')$
$\mu[s] := a$

L'algorithme d'itération de politiques : application

$v(1) = \frac{197}{7}$
$v(2) = 21$
$v(3) = 20$
$v(4) = 0$

μ :
$1 \to a$
$2 \to c$
$3 \to a$

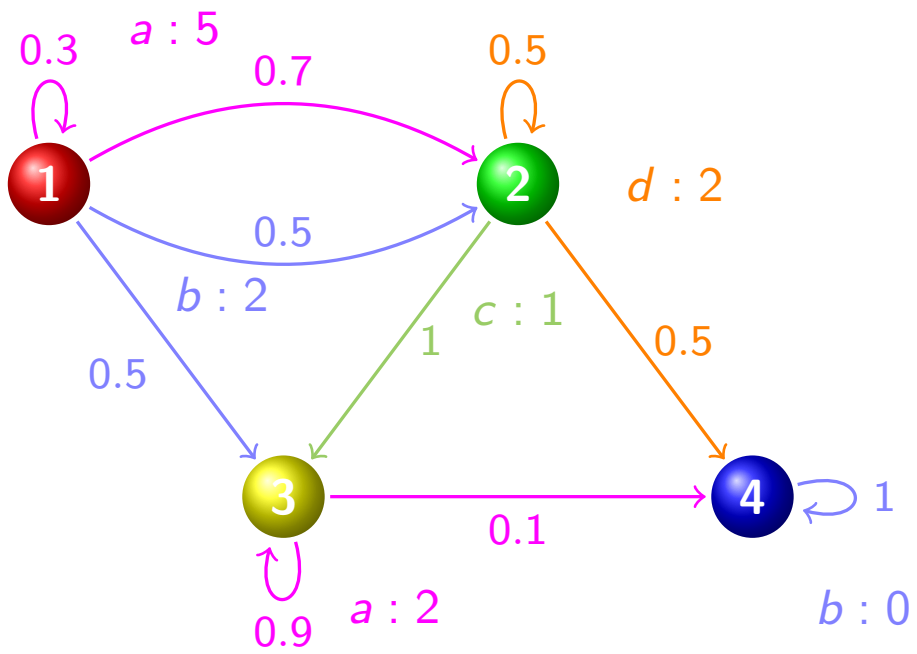


1. Début avec une politique quelconque

L'algorithme d'itération de politiques : application

$v(1) = 14$
$v(2) = 4$
$v(3) = 20$
$v(4) = 0$

μ :
$1 \to b$
$2 \to d$
$3 \to a$

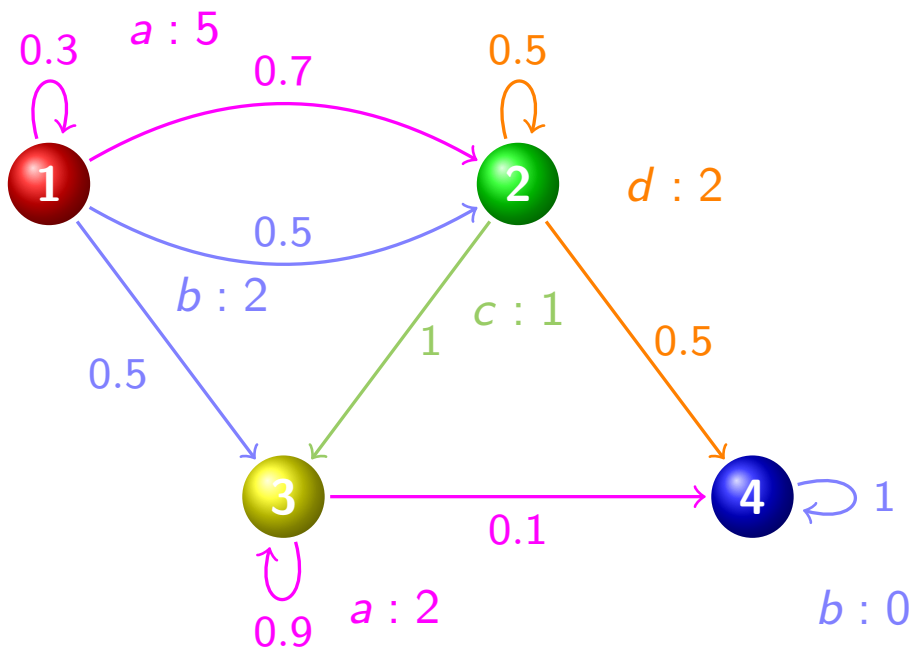


1. Début avec une politique quelconque
2. Amélioration de la politique pour les états 1 et 2

L'algorithme d'itération de politiques : application

$v(1) = \frac{78}{7}$
$v(2) = 4$
$v(3) = 20$
$v(4) = 0$

μ :
$1 \to a$
$2 \to d$
$3 \to a$

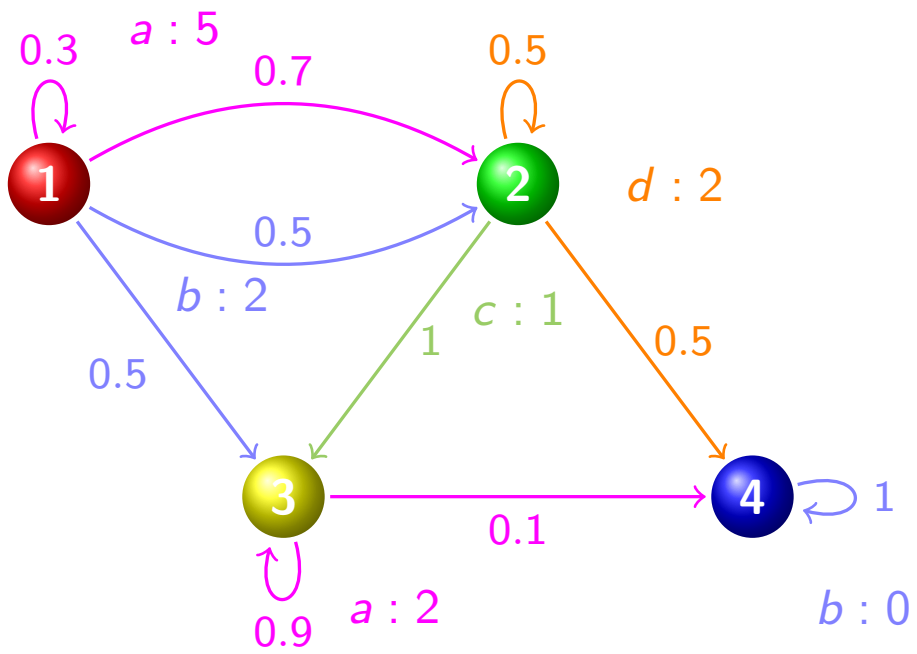


1. Début avec une politique quelconque
2. Amélioration de la politique pour les états 1 et 2
3. Amélioration de la politique pour l'état 1

L'algorithme d'itération de politiques : application

$v(1) = \frac{78}{7}$
$v(2) = 4$
$v(3) = 20$
$v(4) = 0$

μ :
$1 \to a$
$2 \to d$
$3 \to a$

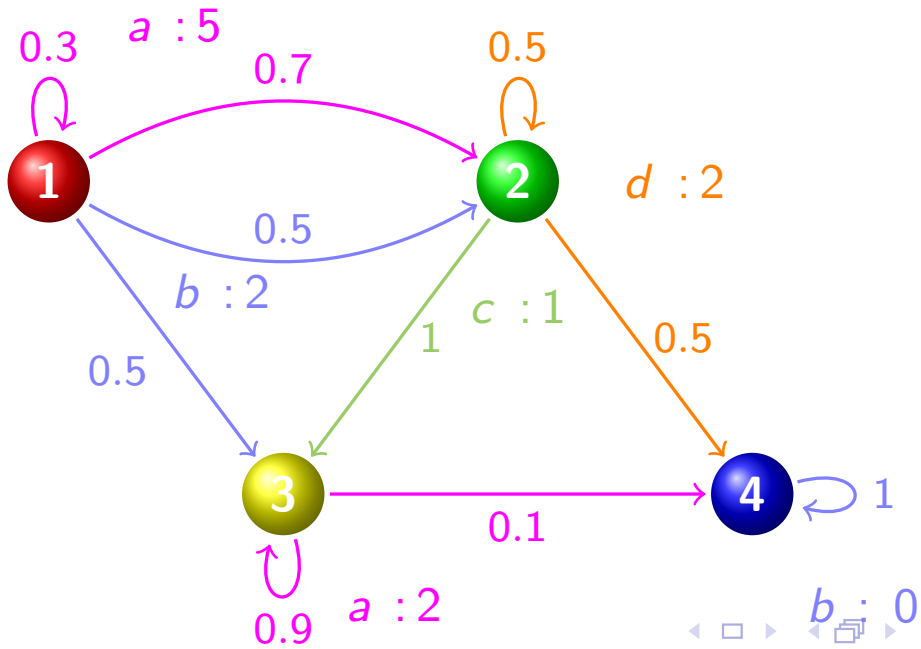


1. Début avec une politique quelconque
2. Amélioration de la politique pour les états 1 et 2
3. Amélioration de la politique pour l'état 1
4. Point fixe : la politique μ est optimale pour $\mathcal{M}$

Plan de l'exposé

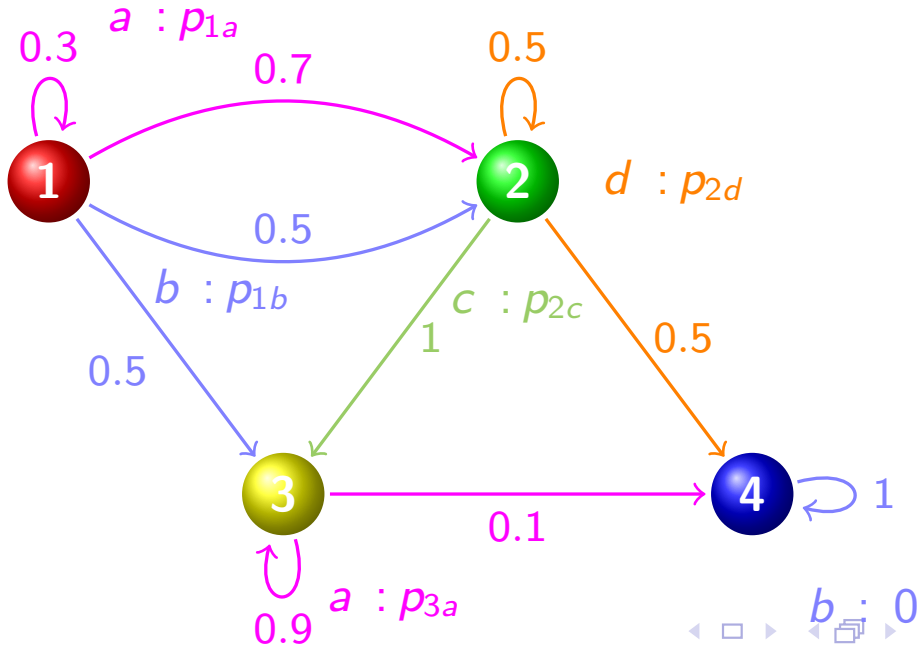
Processus de décision markovien

- Processus de décision markovien
  - Ensemble d'états $S = \{s_1, \ldots, s_n\}$, comprenant un état puits s_n
  - Ensemble A d'actions
  - Fonction de coût w, associant un coût $w(s, a)$ à chaque état s et action a
  - Fonction de probabilité $Prob$, associant une probabilité à chaque arc, telle que la somme des probabilités quittant un état s via l'action a est égale à 1, c'est-à-dire $\sum_{s' \in S} Prob(s, a, s') = 1$



Processus de décision markovien paramétré (PDMP)

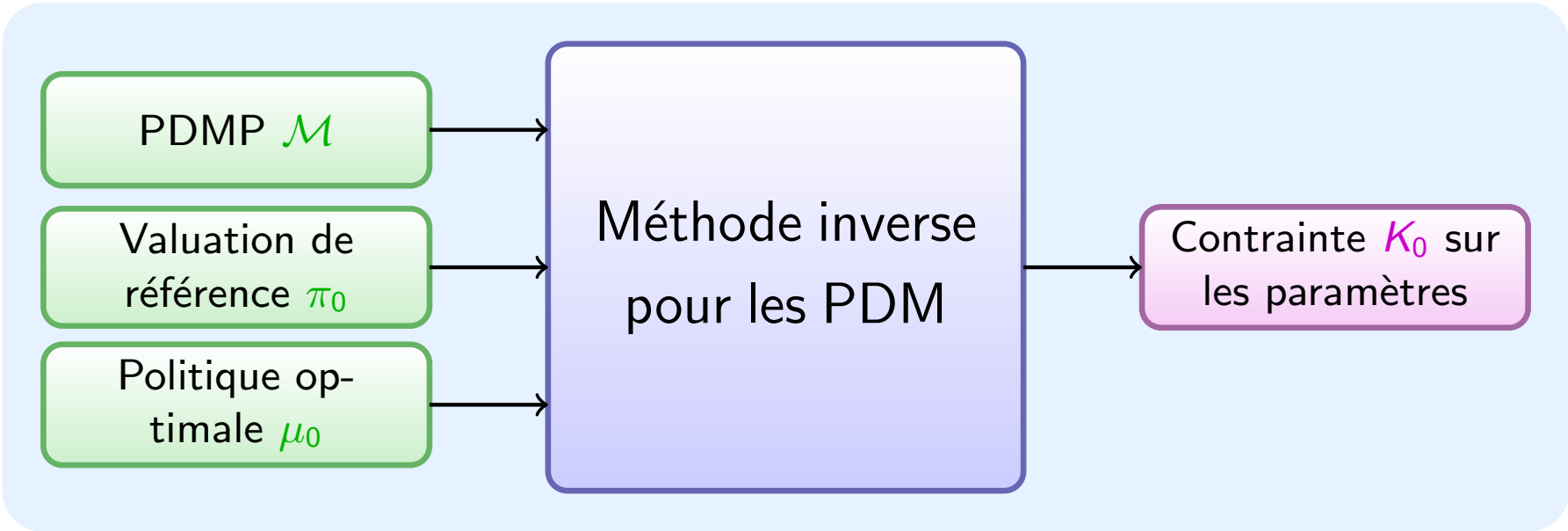
- Processus de décision markovien à coûts paramétriques
  - Ensemble d'états $S = \{s_1, \ldots, s_n\}$, comprenant un état puits s_n
  - Ensemble A d'actions
  - Fonction de coût paramétrique W, associant un coût paramétrique (une constante inconnue) $W(s, a)$ à chaque état s et action a
  - Fonction de probabilité $Prob$, associant une probabilité à chaque arc, telle que la somme des probabilités quittant un état s via l'action a est égale à 1, c'est-à-dire $\sum_{s' \in S} Prob(s, a, s') = 1$



Processus de décision markovien paramétré : remarques

- L'instanciation d'un PDMP $\mathcal{M}$ par une valuation π des paramètres donne un PDM classique (non-paramétrique)
  - Noté $\mathcal{M}[\pi]$
- La paramétrisation d'un PDM en un PDMP est similaire à la paramétrisation des automates temporisés en automates temporisés paramétrés

Entrées et sorties (1/2)

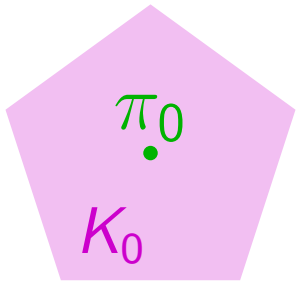


Entrées et sorties (2/2)

- Entrées
  - Un PDMP $\mathcal{M}$
  - Une valuation de référence π_0 de tous les paramètres de $\mathcal{M}$
  - Une politique μ_0 optimale pour $\mathcal{M}[\pi_0]$

Entrées et sorties (2/2)

- Entrées
  - Un PDMP $\mathcal{M}$
  - Une valuation de référence π_0 de tous les paramètres de $\mathcal{M}$
  - Une politique μ_0 optimale pour $\mathcal{M}[\pi_0]$
- Sortie : généralisation
  - Une contrainte K_0 sur les paramètres telle que
    - ★ $\pi_0 \models K_0$, et
    - ★ La politique μ_0 est optimale pour $\mathcal{M}[\pi]$, pour tout $\pi \models K_0$



Le principe général

Étant donnés un PDMP $\mathcal{M}$, une valuation π_0 des paramètres, et une politique μ_0 optimale pour $\mathcal{M}[\pi_0]$

1. Déterminer la fonction de valeur paramétrique pour $\mathcal{M}$ et μ_0, à l'aide d'une version paramétrée de l'algorithme de détermination de valeur
2. Générer des contraintes sur les paramètres de $\mathcal{M}$, en s'inspirant du critère d'optimalité de l'algorithme d'itération de politiques

L'algorithme paramétrique de détermination de valeur

- Adaptation au cas paramétrique de l'algorithme de détermination de valeur *ValueDet*
- Entrées
  - Un PDMP $\mathcal{M} = (S, A, Prob, W)$
  - Une politique μ
- Sortie
  - Une fonction de valeur paramétrique V, associant une valeur paramétrique à chaque état s, c.à.d. le coût paramétrique de s vers l'état puits dans $\mathcal{M}$ restreint à la politique μ

Algorithme (Détermination de valeur paramétrique *P-ValueDet*)

RÉSOUDRE $\{V(s) = W(s, \mu[s]) + \sum_{s' \in S} Prob(s, \mu[s], s') \times V(s')\}_{s \in S \setminus s_n}$

L'algorithme *P-ValueDet* : application

Algorithme (Détermination de valeur paramétrique *P-ValueDet*)

RÉSOUDRE $\{V(s) = W(s, \mu[s]) + \sum_{s' \in S} Prob(s, \mu[s], s') \times V(s')\}_{s \in S \setminus s_n}$

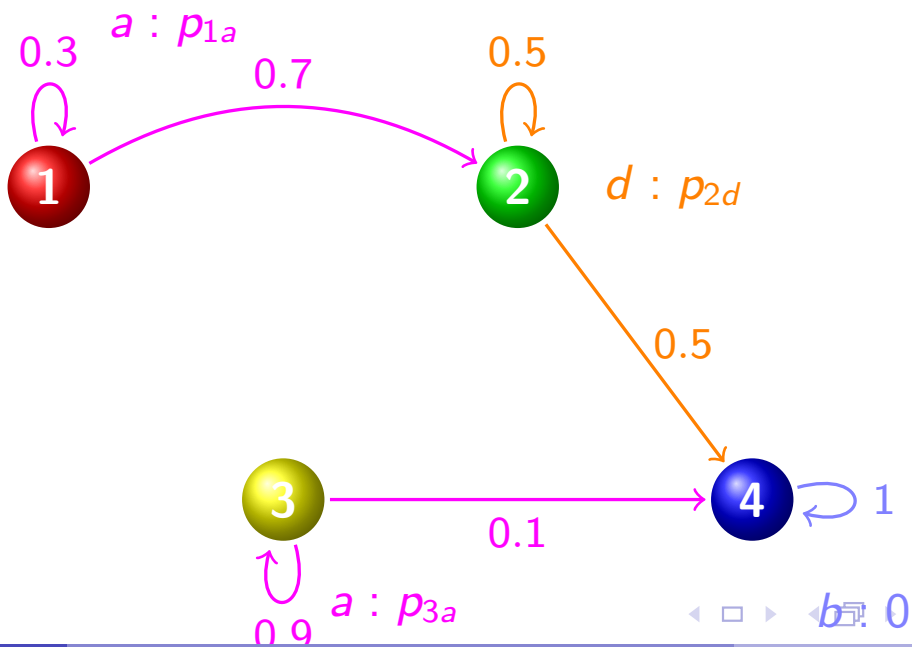
$\mu = \{1 \to a, \quad 2 \to d, \quad 3 \to a\}$

$V(1) = W(1, a) + 0.3 \times V(1) + 0.7 \times V(2)$

$V(2) = W(2, d) + 0.5 \times V(2) + 0.5 \times V(4)$

$V(3) = W(3, a) + 0.9 \times V(3) + 0.1 \times V(4)$

$V(4) = 0$



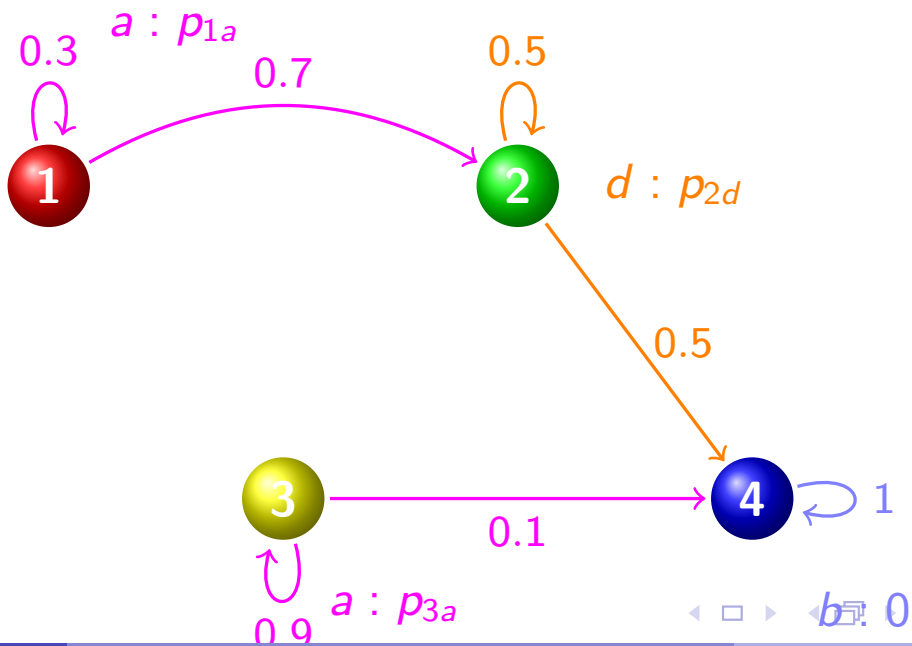
L'algorithme *P-ValueDet* : application

Algorithme (Détermination de valeur paramétrique *P-ValueDet*)

RÉSOUDRE $\{V(s) = W(s, \mu[s]) + \sum_{s' \in S} Prob(s, \mu[s], s') \times V(s')\}_{s \in S \setminus s_n}$

$\mu = \{1 \to a, \quad 2 \to d, \quad 3 \to a\}$

$$
\begin{aligned}
V(1) &= W(1,a) + 0.3 \times V(1) + 0.7 \times V(2) &&= \tfrac{10}{7} \times p_{1a} + 2 \times p_{2d} \\
V(2) &= W(2,d) + 0.5 \times V(2) + 0.5 \times V(4) &&= 2 \times p_{2d} \\
V(3) &= W(3,a) + 0.9 \times V(3) + 0.1 \times V(4) &&= 10 \times p_{3a} \\
V(4) &= 0
\end{aligned}
$$



L'algorithme *MéthodeInverse*

- Entrées
  - Un PDMP $\mathcal{M} = (S, A, Prob, W)$
  - Une valuation π_0 des paramètres
  - Une politique μ_0 optimale pour $\mathcal{M}[\pi_0]$
- Sortie
  - Une contrainte K_0 sur les paramètres solution au problème inverse
- Principe
  - Pour tout état s, pour toute action a, générer une égalité attestant que la politique optimale $\mu_0[s]$ est *meilleure* que a pour s

Algorithme (*MéthodeInverse*)

$V := P\text{-}ValueDet(M, \mu_0)$

$K_0 := True$

POUR TOUT $s \in S \setminus \{s_n\}$ **FAIRE**

  POUR TOUT $a \in e(s)$ *tel que* $a \neq \mu_0[s]$ **FAIRE**

    $K_0 := K_0 \wedge \{W(s,a) + \sum_{s' \in S} Prob(s,a,s')V[s'] \geq V[s]\}$

Propriétés de l'algorithme *MéthodeInverse*

Théorème (Correction [AF09])

Étant donnés un PDMP $\mathcal{M}$, une valuation π_0 des paramètres et une politique μ_0 optimale pour $\mathcal{M}[\pi_0]$, la contrainte K_0 générée par l'algorithme MéthodeInverse est telle que

- $\pi_0 \models K_0$, *et*
- μ_0 *est optimale pour* $\mathcal{M}[\pi]$, *pour tout* $\pi \models K_0$

Théorème (Terminaison et complexité [AF09])

L'algorithme MéthodeInverse termine en temps polynomial.

Application à l'exemple

π_0 :

$p_{1a} = 5$ $\quad p_{1b} = 2$

$p_{2c} = 1$ $\quad p_{2d} = 2$

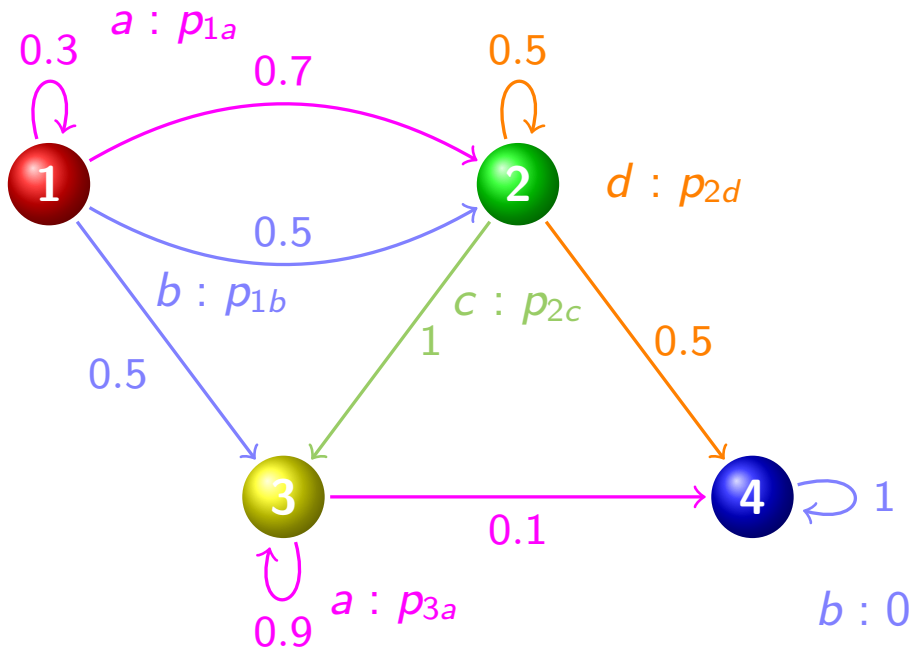
$p_{3a} = 2$

μ_0 :

$1 \rightarrow a$

$2 \rightarrow d$

$3 \rightarrow a$



Application à l'exemple

π_0 :

$p_{1a} = 5$ $\quad p_{1b} = 2$

$p_{2c} = 1$ $\quad p_{2d} = 2$

$p_{3a} = 2$

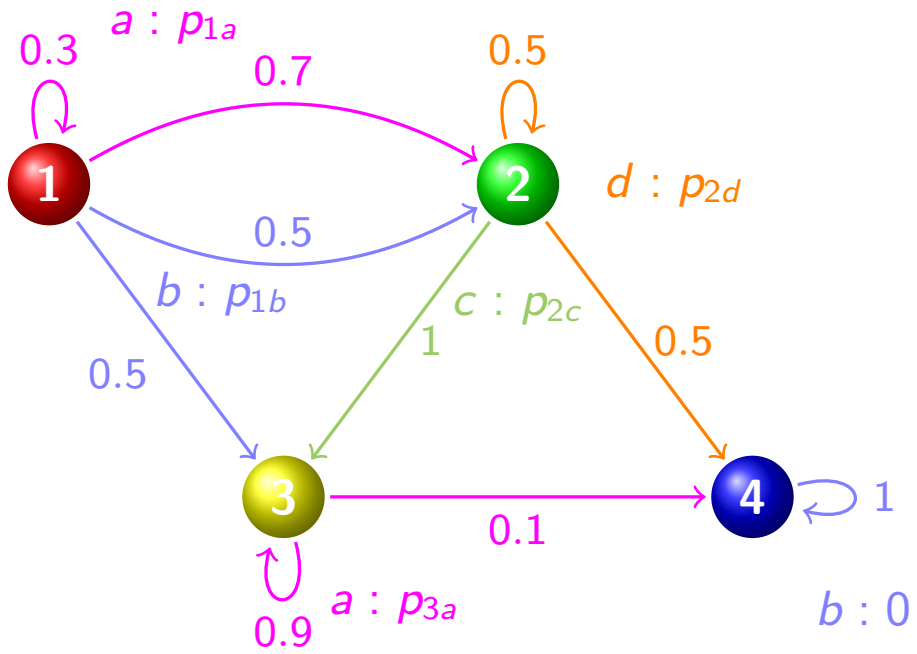
μ_0 :

$1 \to a$

$2 \to d$

$3 \to a$

$$V(1) = \frac{10}{7} \times p_{1a} + 2 \times p_{2d}$$
$$V(2) = 2 \times p_{2d}$$
$$V(3) = 10 \times p_{3a}$$



Application à l'exemple

π_0 :

$p_{1a} = 5$ $\quad p_{1b} = 2$

$p_{2c} = 1$ $\quad p_{2d} = 2$

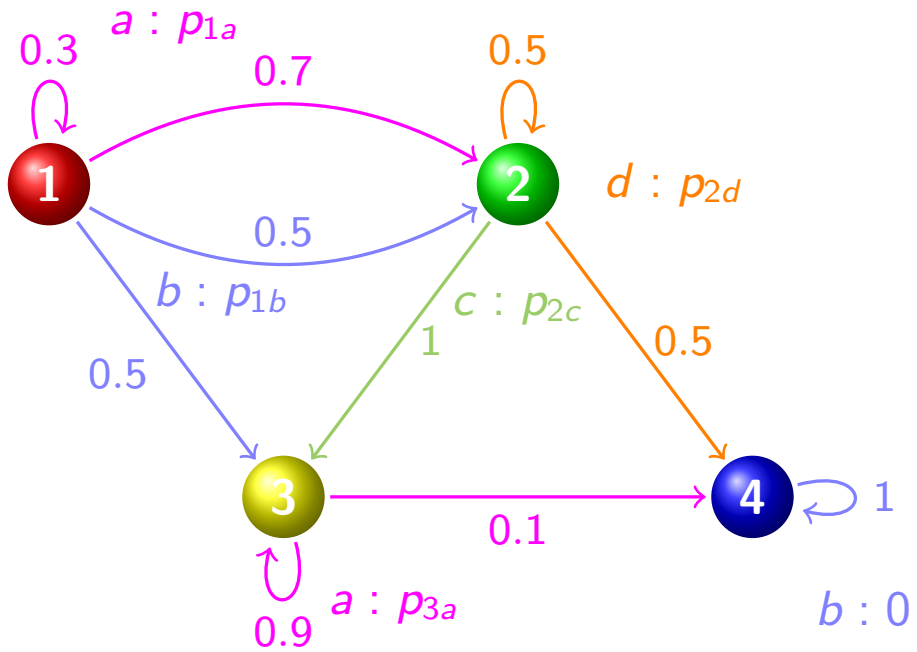
$p_{3a} = 2$

μ_0 :

$1 \rightarrow a$

$2 \rightarrow d$

$3 \rightarrow a$



$$V(1) = \frac{10}{7} \times p_{1a} + 2 \times p_{2d}$$
$$V(2) = 2 \times p_{2d}$$
$$V(3) = 10 \times p_{3a}$$

$K_0 = Vrai$

Application à l'exemple

π_0 :

$p_{1a} = 5$ $\quad p_{1b} = 2$

$p_{2c} = 1$ $\quad p_{2d} = 2$

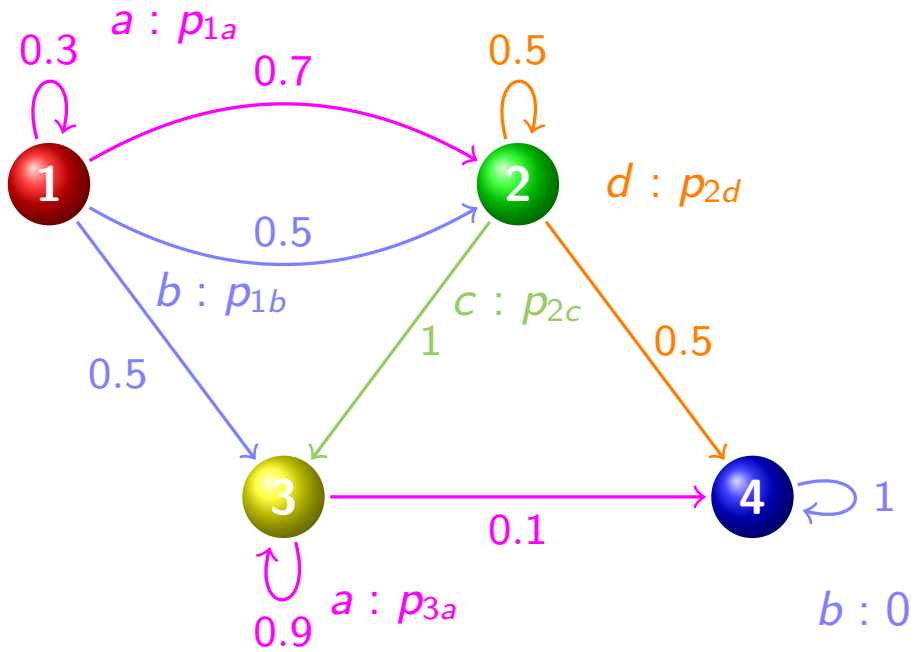
$p_{3a} = 2$

μ_0 :

$1 \to a$

$2 \to d$

$3 \to a$



$$V(1) = \tfrac{10}{7} \times p_{1a} + 2 \times p_{2d}$$
$$V(2) = 2 \times p_{2d}$$
$$V(3) = 10 \times p_{3a}$$

$K_0 =$

$p_{1b} + \frac{1}{2}V(2) + \frac{1}{2}V(3) \geq V(1)$ %% pour 1 et b

Application à l'exemple

π_0 :

$p_{1a} = 5$ $\quad p_{1b} = 2$

$p_{2c} = 1$ $\quad p_{2d} = 2$

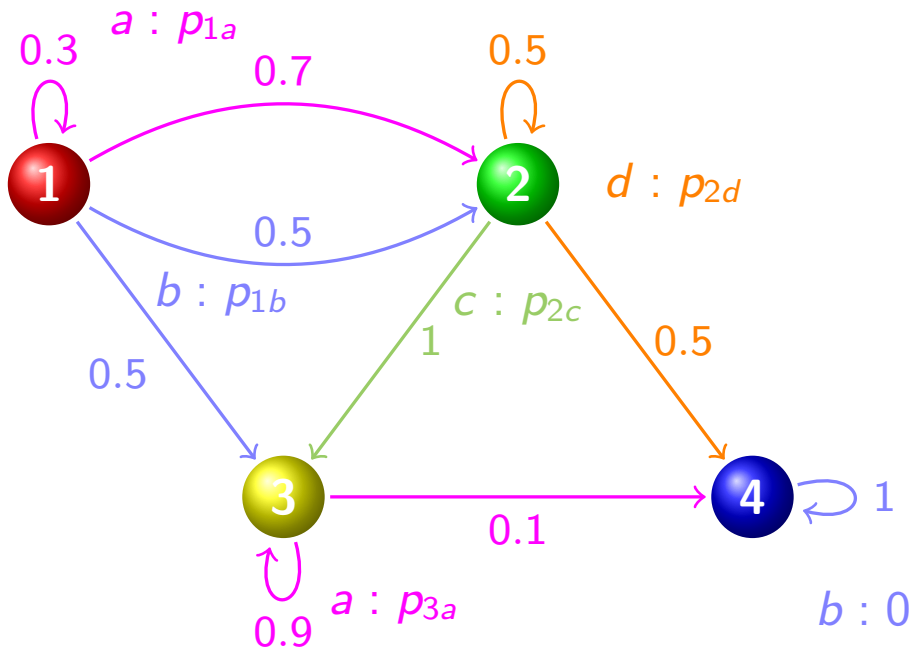
$p_{3a} = 2$

μ_0 :

$1 \to a$

$2 \to d$

$3 \to a$



$$V(1) = \tfrac{10}{7} \times p_{1a} + 2 \times p_{2d}$$
$$V(2) = 2 \times p_{2d}$$
$$V(3) = 10 \times p_{3a}$$

$$K_0 = p_{1b} + \tfrac{1}{2}V(2) + \tfrac{1}{2}V(3) \geq V(1) \quad \%\% \text{ pour 1 et } b$$
$$\wedge \quad p_{2c} + V(3) \geq V(2) \quad \%\% \text{ pour 2 et } c$$

Application à l'exemple

π_0 :

$p_{1a} = 5$ $\quad p_{1b} = 2$

$p_{2c} = 1$ $\quad p_{2d} = 2$

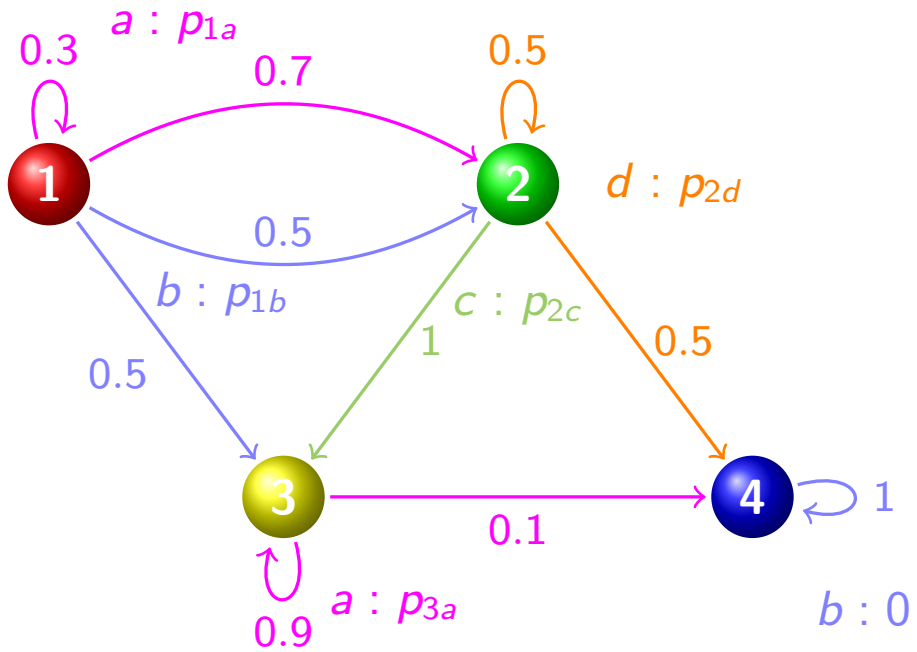
$p_{3a} = 2$

μ_0 :

$1 \rightarrow a$

$2 \rightarrow d$

$3 \rightarrow a$



$$V(1) = \tfrac{10}{7} \times p_{1a} + 2 \times p_{2d}$$
$$V(2) = 2 \times p_{2d}$$
$$V(3) = 10 \times p_{3a}$$

$$K_0 = \quad p_{1b} + 5p_{3a} \geq \tfrac{10}{7} p_{1a} + p_{2d} \quad \wedge \quad p_{2c} + 10p_{3a} \geq 2p_{2d}$$

Application à l'exemple

π_0 :

$p_{1a} = 5$ $\quad p_{1b} = 2$

$p_{2c} = 1$ $\quad p_{2d} = 2$

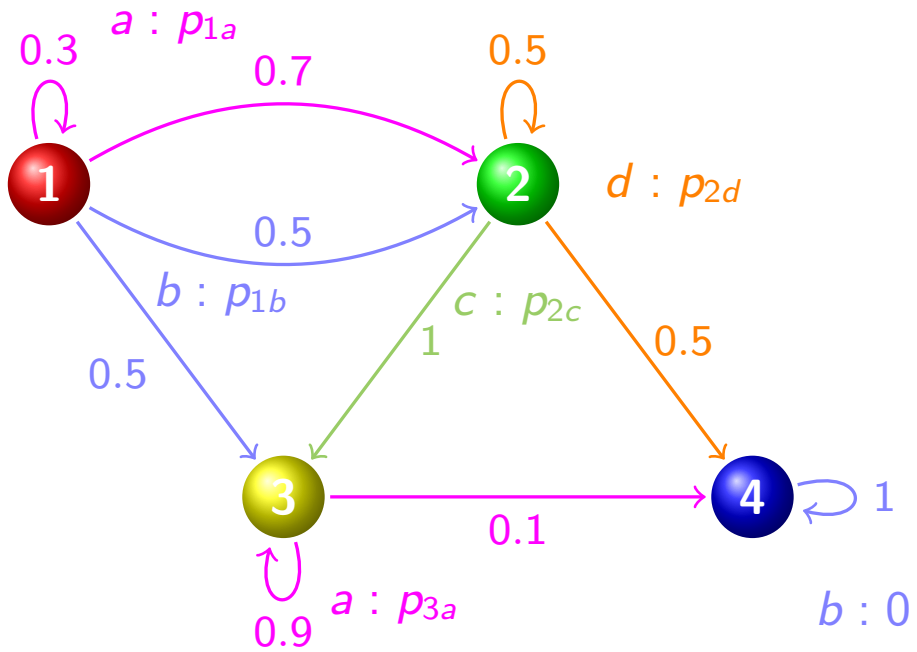
$p_{3a} = 2$

μ_0 :

$1 \to a$

$2 \to d$

$3 \to a$



$V(1) = \frac{10}{7} \times p_{1a} + 2 \times p_{2d}$

$V(2) = 2 \times p_{2d}$

$V(3) = 10 \times p_{3a}$

$K_0 =$

$p_{1b} + 5p_{3a} \geq \frac{10}{7} p_{1a} + p_{2d}$

$\wedge \quad p_{2c} + 10p_{3a} \geq 2p_{2d}$

- Application : maximisation de, par exemple, p_{2d}
  - En instanciant tous les paramètres sauf p_{2d} dans K_0, on obtient l'inégalité $p_{2d} \leq \frac{34}{7}$
  - Si l'on maximise p_{2d} à $\frac{34}{7}$, μ_0 reste optimale

Plan de l'exposé

Implémentation

- ImPrator : programme développé en OCaml
  - ImPrator : « Inverse Method for Policy with Reward AbstracT BehaviOR »
  - 4000 lignes de code
  - 2 hommes-mois de développement
- Fonctionnalités
  - Syntaxe d'entrée très intuitive
  - Résolution du problème direct pour les PDM (non paramétrés)
  - Résolution du problème inverse pour les PDM paramétrés
- ImPrator sera très prochainement disponible sur sa page
  - `http://www.lsv.ens-cachan.fr/~andre/ImPrator`
  - *Coming soon !*

Plan de l'exposé

Remarques finales (1/2)

- Méthode de généralisation
  - Modélisation d'un système avec un processus de décision markovien paramétré $\mathcal{M}$
  - Partant d'une valuation π_0 des paramètres, ainsi que d'une politique μ_0 optimale pour $\mathcal{M}[\pi_0]$, on génère une contrainte K_0 sur les paramètres garantissant que μ_0 est optimale pour $\mathcal{M}[\pi]$, pour tout $\pi \models K_0$
- Avantages
  - Utile pour optimiser les coûts de systèmes, par exemples éléments matériel
  - Puissant même sur de larges systèmes entièrement paramétrés
    - ★ Toutes nos études de cas ont été résolues en moins d'une seconde
- Applications
  - Vérification de systèmes matériel
  - Systèmes temps réel
  - Systèmes avec tout type de coût (consommation énergétique, etc.)

Remarques finales (2/2)

- Autres cadres pour la méthode inverse
  - Automates temporisés paramétrés [ACEF09]
    - ★ Outil IMITATOR [And09a]
  - Graphes orientés valués [And09b]
    - ★ Détermination du plus court chemin
    - ★ Outil INSPEQTOR
  - Algèbre max–plus [AF09]
    - ★ Calcul du circuit moyen maximal dans un graphe orienté valué
    - ★ Outil en cours de développement

- Directions de recherches futures
  - Prouver que la contrainte K_0 générée est maximale
    - ★ Si μ_0 est une politique optimale pour $M[\pi]$, alors $\pi \models K_0$
  - Considérer des PDM à 2 coûts
    - ★ Exemple : (1) consommation énergétique et (2) nombre de requêtes perdues
    - ★ Application : gestion de puissance dynamique (*dynamic power management*) [PBBDM98]

Quelques références I


Étienne André, Thomas Chatain, Emmanuelle Encrenaz, and Laurent Fribourg.
An inverse method for parametric timed automata.
International Journal of Foundations of Computer Science, 2009.
To appear.


É. André and L. Fribourg.
An inverse method for policy-iteration based algorithms.
In *INFINITY '09*, August 2009.


Étienne André.
IMITATOR : A tool for synthesizing constraints on timing bounds of timed automata.
In *ICTAC'09*, LNCS. Springer, August 2009.
To appear.


Étienne André.
Une méthode inverse pour les plus courts chemins.
In *ETR '09*, Paris, France, September 2009.


R. Bellman.
A Markov decision process.
Journal of Mathematical Mechanics, 6 :679–684, 1957.

Quelques références II


R. A. Howard.
Dynamic Programming and Markov Processes.
John Wiley and Sons, Inc., 1960.


J. Kemeny, H. Mirkil, J. Snell, and G. Thompson.
Finite mathematical structures.
Prentice-Hall, Englewood Cliffs, N.J., 1959.


G. A. Paleologo, L. Benini, A. Bogliolo, and G. De Micheli.
Policy optimization for dynamic power management.
In *DAC '98*, pages 182–187, New York, NY, USA, 1998. ACM.