

# Synthèse multi-objectifs dans les processus décisionnels de Markov

---

**Florent Delgrange**

UMONS

Faculté des Sciences

Mab2 Science Informatique

# Table des matières

## 1. SP-G

### 1.1 Motivations

### 1.2 Définition

### 1.3 Algorithme

### 1.4 Exemple

## 2. SSP-WE

### 2.1 Motivations

### 2.2 Définition

### 2.3 Algorithme

### 2.4 Exemple

## 3. SSP-PQ

## 3.1 MDP

### multidimensionnels

### 3.2 Motivations

### 3.3 Définition

### 3.4 Algorithme

## 4. G-SR

### 4.1 Définition

### 4.2 Préliminaires

### 4.3 MOLP

### 4.4 Stratégie

### 4.5 Exemple dans storm

## Borne supérieure stricte dans un MDP

Soit  $\mathcal{M} = (S, A, \Delta, w)$ , un MDP,  $s \in S$ , un état de  $\mathcal{M}$  et  $T \subseteq S$ , un sous-ensemble d'états cibles.

On va définir une stratégie  $\sigma$  qui **garantit d'atteindre  $T$  depuis  $s$  avec un coût inférieur à un seuil  $l$**  :

$$\rightsquigarrow \forall \pi \in Paths^\sigma(s), TS^T(\pi) \leq l$$

$$\equiv \mathbb{P}_s^\sigma(\Diamond_{\leq l} T) = 1$$

On veut donc assurer une **borne supérieure stricte** lors de l'accessibilité à  $T$ .

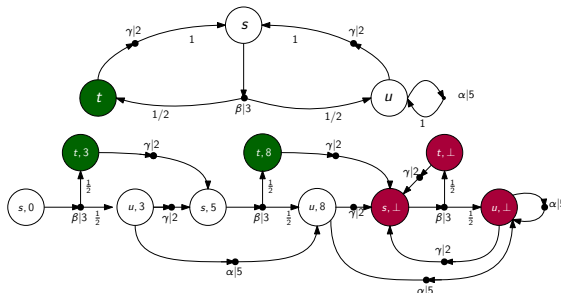
# Borne supérieure stricte dans un MDP

## Remarque

- $\forall \pi \in \text{Paths}^\sigma(s), TS^T(\pi) \leq I$
- $\mathbb{P}_s^\sigma(\Diamond_{\leq I} T) = 1$

$w : A \rightarrow \mathbb{N}_0 \implies$  Ces deux propositions sont équivalentes !

En effet, le problème SSP-P induit un **dépliage** de  $\mathcal{M}$ , dont le graphe sous-jacent est un DAG (si on ne considère pas les états terminaux).



# Problème du plus court chemin dans un jeu

Assurer une borne supérieure stricte lors de l'accessibilité à  $T$

↪ Problème **SP-G** (*Shortest path game problem*)

À chaque étape,

- **Joueur 1** : choisit l'action par stratégie
- **Joueur 2** : choisit le successeur  $s'$ , i.e., l'état qui mène au pire cas en terme de somme tronquée

# Problème du plus court chemin dans un jeu

## Definition (SP-G)

Soient  $\mathcal{M} = (S, A, \Delta, w)$  un MDP,  $s \in S$ , un état de  $\mathcal{M}$ ,  $T \subseteq S$ , un sous-ensemble d'états cibles et un seuil  $l \in \mathbb{N}$ .

Le problème **SP-G** consiste à décider s'il existe une stratégie  $\sigma$  pour laquelle

$$\begin{aligned} \forall \pi \in \text{Paths}^\sigma(s), TS^T(\pi) \leq l \\ \equiv \mathbb{P}_s^\sigma(\Diamond_{\leq l} T) = 1 \end{aligned}$$

- peut être décidé en temps polynomial
- stratégie sans mémoire

# Problème du plus court chemin dans un jeu

## Hypothèses

$$w : A \rightarrow \mathbb{N}_0$$

- Les coûts sont strictement positifs
- ↪ La stratégie ne choisit pas d'action qui pourrait mener à un cycle pendant la minimisation de  $TS^T$ 
  - pas d'intérêt car poids strictement positifs
- ↪  $TS^T = \infty$  si les cycles ne peuvent pas être évités

## SP-G : Programmation dynamique

### Algorithme

$\mathbb{C}(s, i)$  est le plus court chemin jusque  $T$  depuis  $s$ , après  $i$  étapes,  
 $\forall 0 \leq i \leq n, n = |S|$  (pas de cycle).

### Initialisation :

- $\forall t \in T, \mathbb{C}(t, 0) = 0$
- $\forall s \in S \setminus T, \mathbb{C}(s, 0) = \infty$

Soit  $k \in \mathbb{N}$  tel que  $0 < k \leq n$ . Supposons que  $\mathbb{C}(s, k-1)$  a déjà été calculé pour tout  $s \in S$ . Alors, pour tout  $s \in S$ ,

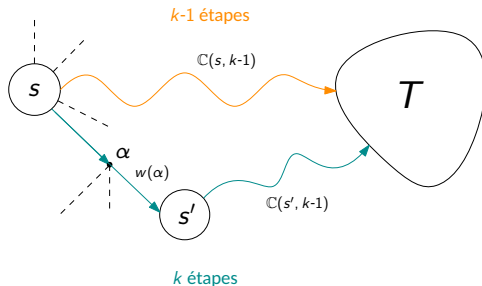
$$\mathbb{C}(s, k) = \min\{\mathbb{C}(s, k-1), \underbrace{\min_{\alpha \in A(s)} \max_{s' \in \text{Succ}(s, \alpha)} w(\alpha) + \mathbb{C}(s', k-1)}_{\text{l'adversaire choisit le pire successeur}}\}$$



# SP-G : Programmation dynamique

## Algorithme

$$\mathbb{C}(s, k) = \min\{\mathbb{C}(s, k-1), \min_{\alpha \in A(s)} \max_{s' \in \text{Succ}(s, \alpha)} w(\alpha) + \mathbb{C}(s', k-1)\}$$

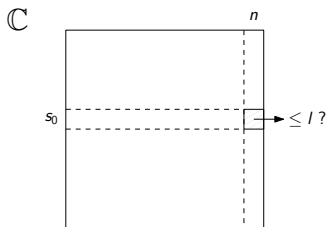


# SP-G : Programmation dynamique

## Algorithmes

La stratégie  $\sigma$  existe s'il est possible d'atteindre  $T$  depuis  $S_0$  avec une longueur de chemin d'au plus  $l$  :

$$\mathbb{C}(s_0, n) \leq l \iff \exists \sigma, \mathbb{P}_{s_0}^{\sigma}(\Diamond_{\leq l} T) = 1$$



# SP-G : Programmation dynamique

## Algorithmes

### Résumé :

- $\mathbb{C}(s, k)$  correspond au plus court chemin de  $s \in S$  à  $T$  après **au plus**  $k$  étapes, i.e., après avoir traversé **au plus**  $k$  états.
- À chaque étape, le plus court chemin résulte du choix des actions minimisant, à chaque étape, le coût de la transition menant au *pire* successeur (i.e., celui dont le chemin pour atteindre  $T$  est le plus long) est choisi.
- Cycle depuis un état  $\implies$  on ne peut pas atteindre  $T$
- $\forall k \in \mathbb{N}, \mathbb{C}(s, k-1) \geq \mathbb{C}(s, k)$  pour tout  $s \in S$
- $\mathbb{C}(s, n) = \min_k \mathbb{C}(s, k)$  pour tout  $s \in S$

## SP-G

### *Construction de la stratégie*

Soit  $s \in S$ . Par le fait que

- $\forall k \in \mathbb{N}, \mathbb{C}(s, k-1) \geq \mathbb{C}(s, k)$  pour tout  $s \in S$
- $\mathbb{C}(s, n) = \min_k \mathbb{C}(s, k)$  pour tout  $s \in S$

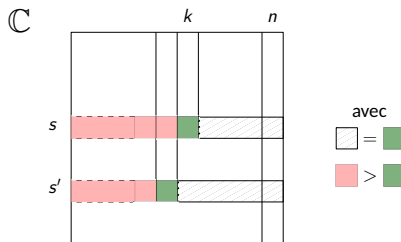
On peut construire une stratégie sans mémoire en se basant sur les résultats de  $\mathbb{C}(s, n)$  de chaque  $s \in S$  :

$$\sigma : S \rightarrow A, s \mapsto \arg \min_{\alpha \in A(s)} \max_{s' \in \text{Succ}(s, \alpha)} w(\alpha) + \mathbb{C}(s', n)$$

# SP-G

## Construction de la stratégie : intuition

Soient  $k \in \mathbb{N}$ ,  $s \in S \setminus T$ . Supposons que  $\mathbb{C}(s, n) \neq \infty$ .



- $\exists k \leq n$  tel que  $\mathbb{C}(s, n) = \mathbb{C}(s, k) < \mathbb{C}(s, k - i) \quad \forall i \in \{0, \dots, k\}$   
 $\rightsquigarrow s$  atteint  $T$  en  $k$  étapes.
- $\exists s' \in Succ(s)$  et  $\exists \alpha \in A(s)$  tels que  
 $\mathbb{C}(s, k) = \mathbb{C}(s', k - 1) + w(\alpha) = \mathbb{C}(s', n) + w(\alpha)$   
 $\rightsquigarrow s$  accède à  $T$  via  $s'$ , qui atteint  $T$  en  $k - 1$  étapes.

# SP-G

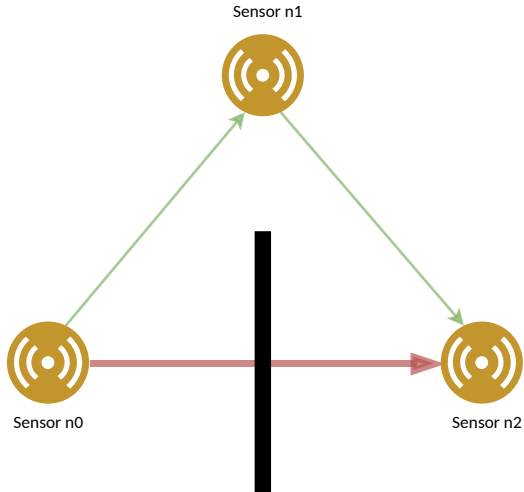
## Note

- poids négatifs : possible de résoudre en temps pseudo-polynomial
- multi-dimensionnel + poids négatifs : indécidable

## Exemple

# Exemple

*Communication entre noeuds dans un réseau de capteurs*



## Exemple

### *Communication entre noeuds dans un réseau de capteurs*

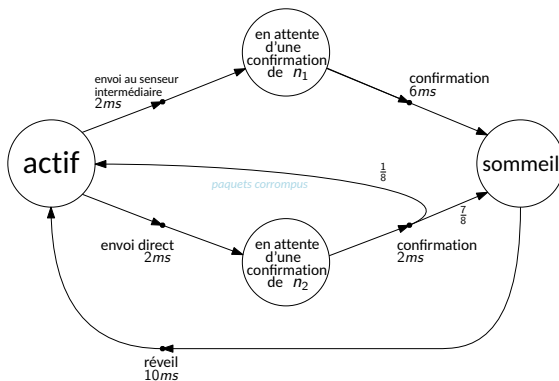
- Un mur sépare  $n_0$  et  $n_2$
- Communication directe  $n_0 \rightarrow n_2$ 
  - plus rapide que de passer par un noeud intermédiaire
  - nécessite plus d'énergie
  - risque de corruption des paquets envoyés (bruit)
- Communication indirecte :  $n_0 \rightarrow n_1 \rightarrow n_2$ 
  - plus lent ( $n_1$  doit attendre la confirmation de réception du paquet par  $n_2$  et  $n_0$  doit attendre la confirmation de  $n_1$ )
  - consommation d'énergie normale
  - risque de perte de paquet négligeable



## Exemple

## Exemple

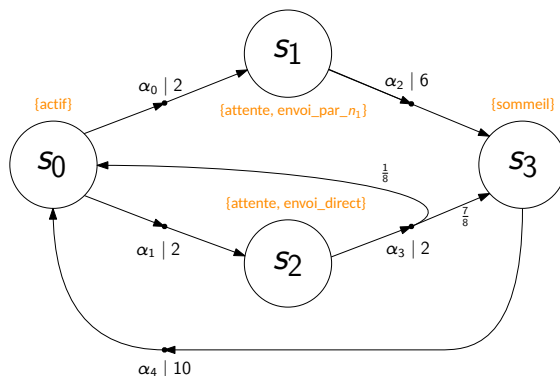
Communication entre noeuds dans un réseau de capteurs



## Exemple

## Exemple

*Communication entre noeuds dans un réseau de capteurs*

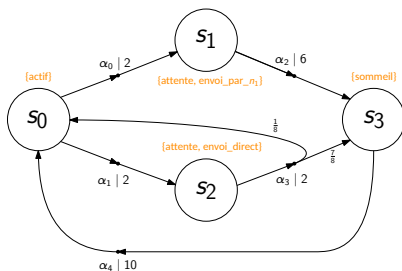


## Exemple

## Exemple

Communication entre noeuds dans un réseau de capteurs

Dutty cycle : 12ms  $\rightsquigarrow \exists \sigma, \mathbb{P}_{s_0}^\sigma (\Diamond_{\leq 12} \text{sommeil}) = 1 ?$



$$k=0 \quad - \quad \mathbb{C}(s_0, 0) = \mathbb{C}(s_1, 0) =$$

$$\mathbb{C}(s_2, 0) = \infty$$

$$- \quad \mathbb{C}(s_3, 0) = 0$$

$$k=1 \quad - \quad \mathbb{C}(s_0, 1) = \infty$$

$$- \quad \mathbb{C}(s_1, 1) = 6 + 0 = 6$$

$$- \quad \mathbb{C}(s_2, 1) = 2 + \infty = \infty$$

$$k=2 \quad - \quad \mathbb{C}(s_0, 2) = 2 + 6 = 8$$

$$k=3 \quad - \quad \mathbb{C}(s_2, 3) = 2 + 8 = 10$$

# Table des matières

## 1. SP-G

### 1.1 Motivations

### 1.2 Définition

### 1.3 Algorithme

### 1.4 Exemple

## 2. SSP-WE

### 2.1 Motivations

### 2.2 Définition

### 2.3 Algorithme

### 2.4 Exemple

## 3. SSP-PQ

## 3.1 MDP

### multidimensionnels

### 3.2 Motivations

### 3.3 Définition

### 3.4 Algorithme

## 4. G-SR

### 4.1 Définition

### 4.2 Préliminaires

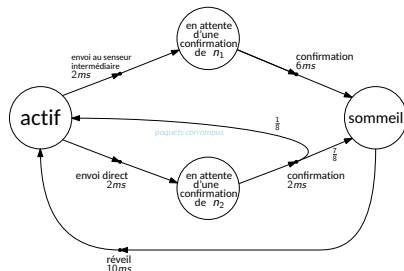
### 4.3 MOLP

### 4.4 Stratégie

### 4.5 Exemple dans storm

## Au delà du pire cas...

- On souhaite assurer simultanément un **seuil de pire cas** et une **bonne espérance**



Quelle est l'espérance optimale du temps d'envoi d'informations au senseur  $n_2$  qui assure de respecter le *duty cycle* du senseur  $n_0$  (12ms) ?

# Espérance sous un pire cas

## Definition (SSP-WE)

Soient  $\mathcal{M} = (S, A, \Delta, w)$ , un MDP à une dimension (i.e., tel que  $w : A \rightarrow \mathbb{N}_0$ ), un état initial  $s \in S$ , un sous-ensemble d'états cibles  $T \subseteq S$  et deux seuils  $l_1, l_2 \in \mathbb{N}$ . Le problème consiste à décider s'il existe une stratégie  $\sigma$  pour laquelle

- $\forall \pi \in \text{Paths}^\sigma(s), TS^T(\pi) \leq l_1 \quad \equiv \quad \mathbb{P}_s^\sigma(\Diamond_{\leq l_1} T) = 1$
- $\mathbb{E}_s^\sigma(\Diamond T) \leq l_2$

- décidé en temps **pseudo-polynomial**
- NP-hard
- mémoire pseudo-polynomiale lors de la construction de la stratégie

# SSP-WE

## Algorithme

1. Construire  $\mathcal{M}_{l_1}$ , le MDP  $\mathcal{M}$  déplié jusqué  $l_1$
2. Calculer,  $\mathbb{A}$ , l'ensemble des actions possibles de l'*attracteur* de  $T' = \{(t, v) \in S_{l_1} \mid t \in T \text{ et } v \leq l_1\}$ 
  - ↪ pour chaque état  $(s, v) \in S_{l_1}$ ,  $\mathbb{A}(s, v)$  est l'ensemble des actions  $\alpha \in A(s)$  qui assurent à  $(s, v)$  d'atteindre  $T'$  dans  $\mathcal{M}_{l_1}$ , quelle que soit l'issue de l'évolution du système par l'*adversaire* (i.e., l'incertitude lié aux probabilités).
3. Construire  $\mathcal{M}_{l_1}^{\mathbb{A}}$ , le MDP déplié limité à l'attracteur de  $T'$ 
  - ↪ on supprime les états  $(s, v)$  de  $\mathcal{M}_{l_1}$  tels que  $\mathbb{A}(s, v) = \emptyset$
4. Résoudre le problème SSP-E sur  $\mathcal{M}_{l_1}^{\mathbb{A}}$

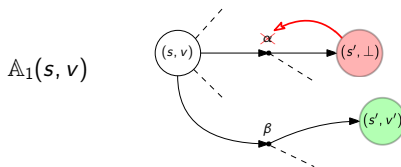
# SSP-WE

## Algorithme

$\mathbb{A}(s, v)$  peut être calculé récursivement :

- **Idée** :  $\mathbb{A}_i(s, v)$  correspond à l'ensemble des actions garantissant au système d'évoluer vers un état sûr (i.e., non-terminal) après  $i$  étapes.
- $\mathbb{A}_0(s, v) = A(s)$  si  $v \leq l$   
 $\mathbb{A}_0(s, v) = \emptyset$  si  $v = \perp$
- Soit  $i \in \mathbb{N}$ . On suppose que  $\mathbb{A}_i(s, v)$  a été calculé pour tout  $(s, v) \in S_{l_1}$ . Alors,

$$\mathbb{A}_{i+1}(s, v) = \{\alpha \in \mathbb{A}_i(s, v) \mid \forall (s', v') \in Succ((s, v), \alpha), \mathbb{A}_i(s', v') \neq \emptyset\}$$





# SSP-WE

## Algorithme

- La taille de l'ensemble  $\mathbb{A}_i$  est croissante en  $i$ , i.e.,  
 $|\mathbb{A}_i| \geq |\mathbb{A}_{i+1}| \quad \forall i \in \mathbb{N}$  par construction de  $\mathbb{A}$
- Le dépliage du MDP résulte en un DAG
  - $\Rightarrow$  pas de cycle !
  - $\Rightarrow$  le nombre d'itérations  $i$  est donc borné par  $v$  (= pire cas  $\rightarrow$  toutes les actions ont un coût de 1)
  - $\Rightarrow$  le temps de construction de  $\mathbb{A}$  est donc polynomial en la taille du MDP déplié  $\mathcal{M}_{I_1} \rightsquigarrow \mathcal{O}(v \times |S_{I_1}|)$

## SSP-WE

### *Construction de la stratégie*

$\sigma$  est la stratégie à mémoire finie résultante de la résolution du problème SSP-E sur le MDP  $\mathcal{M}_{I_1}^{\mathbb{A}}$  ( $\sigma$  est sans mémoire sur  $\mathcal{M}_{I_1}^{\mathbb{A}}$ ).

↪ complexité pseudo-polynomiale

↪ NP-difficile (pas de temps polynomial à moins que  $P = NP$ )

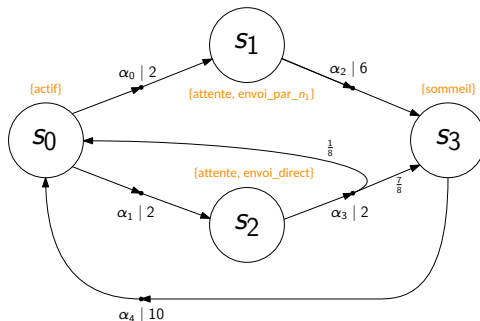
## Exemple

## SSP-WE

## Exemple

$$\stackrel{?}{\exists \sigma} \mathbb{E}_{s_0}^{\sigma} (\Diamond \text{ sommeil}) \leq l \wedge \mathbb{P}_{s_0}^{\sigma} (\Diamond_{\leq 12} \text{ sommeil}) = 1$$

Avec  $l$ , l'espérance minimale pour laquelle le noeud  $n_0$  est assuré d'envoyer les données au noeud  $n_2$  en moins de  $12ms$  (afin d'assurer un *dutty cycle* de  $12ms$ ).

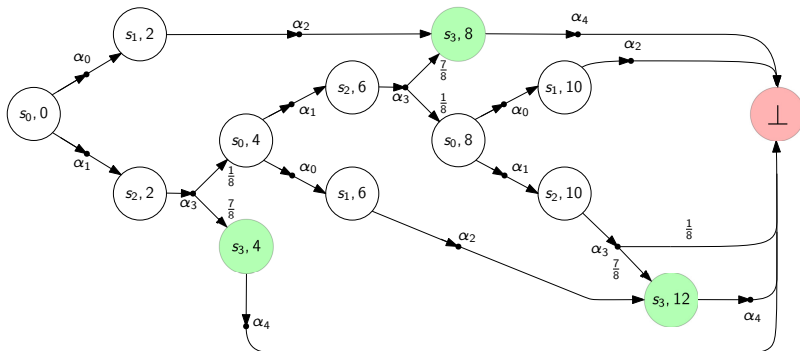


Exemple

## SSP-WE

## Exemple

$$? \exists \sigma \mathbb{E}_{s_0}^{\sigma} (\Diamond \text{ sommeil}) \leq 1 \wedge \mathbb{P}_{s_0}^{\sigma} (\Diamond_{\leq 12} \text{ sommeil}) = 1$$

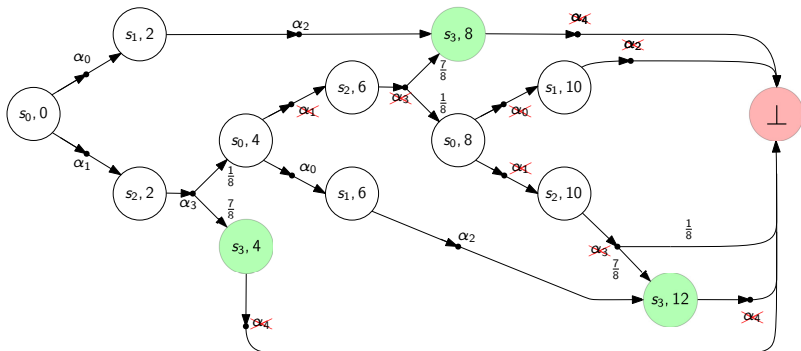


Exemple

## SSP-WE

## Exemple

$$? \exists \sigma \mathbb{E}_{s_0}^{\sigma} (\Diamond \text{ sommeil}) \leq 1 \wedge \mathbb{P}_{s_0}^{\sigma} (\Diamond_{\leq 12} \text{ sommeil}) = 1$$

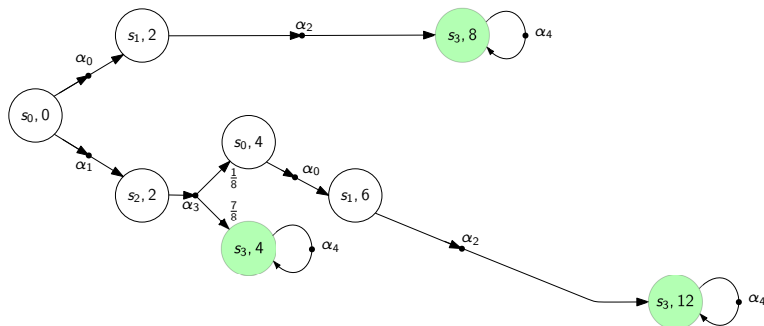


Exemple

## SSP-WE

Exemple

$$\rightsquigarrow \mathbb{E}_{s_0,0}^{\min}(\Diamond\{(s_3, v) \mid v \leq 12\}) = ?$$

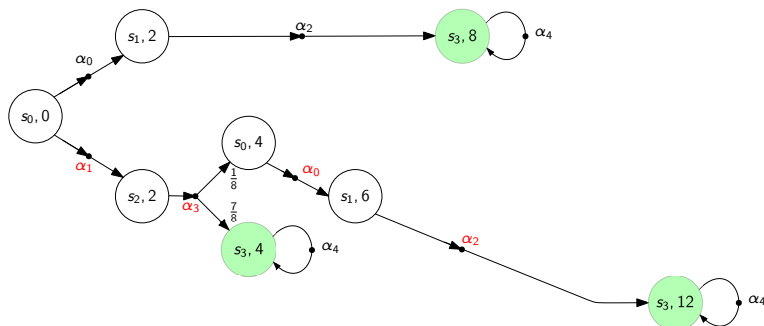


Exemple

## SSP-WE

## Exemple

$$\rightsquigarrow \mathbb{E}_s^{\min}(\Diamond\{(s_3, v) \mid v \leq 12\}) = \frac{7}{8} \cdot 4 + \frac{1}{8} \cdot 12 = 5$$

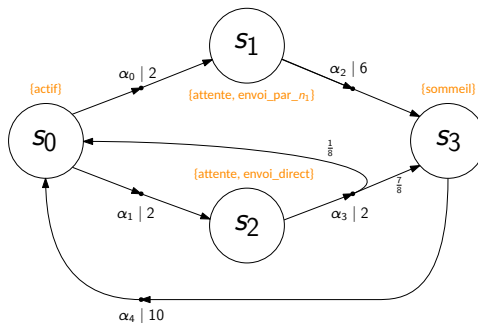


Exemple

## SSP-WE

## Exemple

$$\mathbb{E}_{S_0}^{\sigma}(\Diamond \text{ sommeil}) \leq 5 \wedge \mathbb{P}_{S_0}^{\sigma}(\Diamond_{\leq 12} \text{ sommeil}) = 1$$



- **Stratégie optimale  $\sigma$**  : tester une fois un envoi direct et passer par le noeud  $n_1$  si l'envoi direct est un échec.



Exemple

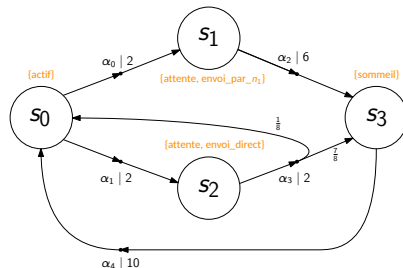
## SSP-WE

## Exemple dans Storm

```

1  mdp
2
3  module sensors_transmission
4
5  s: [0..3] init 0;
6
7  [alpha0] s=0 -> 1 : (s'=1);
8  [alpha1] s=0 -> 1 : (s'=2);
9  [alpha2] s=1 -> 1 : (s'=3);
10 [alpha3] s=2 -> 0.125 : (s'=0) + 0.875 : (s'=3);
11 [alpha4] s=3 -> 1 : (s'=0);
12
13 endmodule
14
15 label "active" = s=0;
16 label "sleep" = s=3;
17 label "waiting" = s=1 | s=2;
18 label "intermediate" = s=1;
19 label "direct" = s=2;
20
21 rewards "time"
22 [alpha0] true : 2;
23 [alpha1] true : 2;
24 [alpha2] true : 6;
25 [alpha3] true : 2;

```



## Exemple

## SSP-WE

*Exemple dans Storm*

$$\begin{aligned}
& \stackrel{?}{\exists} \sigma \mathbb{E}_{s_0}^{\sigma} (\Diamond \text{ sommeil}) \leq 5 \wedge \mathbb{P}_{s_0}^{\sigma} (\Diamond_{\leq 12} \text{ sommeil}) = 1 \\
& \equiv_{\text{PRCTL}} s_0 \models \stackrel{?}{multi}(\mathcal{E}_{\leq 5}^{\min}(\Diamond \text{ sommeil}), \mathcal{P}_{=1}^{\max}(\Diamond_{\leq 12} \text{ sommeil}))
\end{aligned}$$

```
storm --prism resources/sensors.prism --prop "multi(Rmin<=5 [F \"sleep\"], Pmax>=1 [F{\"time\"}<=12 \"sleep\"])"
Storm 1.2.0
```

```
Command line arguments: --prism resources/sensors.prism --prop multi(Rmin<=5 [F "sleep"], Pmax>=1 [F{"time"}<=12 "sleep"])
```

```
Time for model construction: 0.305s.
```

```
-----
Model type: MDP (sparse)
States: 4
Transitions: 6
Choices: 5
Reward Models: time
State Labels: 3 labels
  * deadlock -> 0 item(s)
  * init -> 1 item(s)
  * sleep -> 1 item(s)
Choice Labels: none
-----
```

```
Model checking property multi(R[exp]min<=5 [F "sleep"], Pmax>=1 [true Urew{"time"}<=12 "sleep"]) ...
Result (for initial states): true
Time for model checking: 0.038s.
```

## Exemple

## SSP-WE

*Exemple dans Storm (requête)*

$$\sigma \mid \mathbb{P}_{s_0}^{\sigma} (\Diamond_{\leq 12} \text{sommeil}) = 1 \quad \mathbb{E}_{s_0}^{\sigma} (\Diamond \text{sommeil})$$

$$\equiv_{\text{PRCTL}} s_0 \models \text{multi}(\mathcal{E}_{=?}^{\min}(\Diamond \text{sommeil}), \mathcal{P}_{=1}^{\max}(\Diamond_{\leq 12} \text{sommeil}))$$

```
storm --prism resources/sensors.prism --prop "multi(Rmin=? [F \"sleep\"], Pmax>=1 [F{\"time\"}<=12 \"sleep\"])"
Storm 1.2.0
```

Command line arguments: --prism resources/sensors.prism --prop multi(Rmin=? [F "sleep"], Pmax>=1 [F{"time"}<=12 "sleep"])

Time for model construction: 0.348s.

```
-----
Model type: MDP (sparse)
States: 4
Transitions: 6
Choices: 5
Reward Models: time
State Labels: 3 labels
  * deadlock -> 0 item(s)
  * init -> 1 item(s)
  * sleep -> 1 item(s)
Choice Labels: none
-----
```

```
Model checking property multi(R[exp]min=? [F "sleep"], Pmax>=1 [true Urew{"time"}<=12 "sleep"]) ...
Result (for initial states): 5
Time for model checking: 0.035s.
```

# Table des matières

## 1. SP-G

1.1 Motivations

1.2 Définition

1.3 Algorithme

1.4 Exemple

## 2. SSP-WE

2.1 Motivations

2.2 Définition

2.3 Algorithme

2.4 Exemple

## 3. SSP-PQ

## 3.1 MDP

multidimensionnels

3.2 Motivations

3.3 Définition

3.4 Algorithme

## 4. G-SR

4.1 Définition

4.2 Préliminaires

4.3 MOLP

4.4 Stratégie

4.5 Exemple dans storm

# MDP multidimensionnels

## Definition (MDP multidimensionnel)

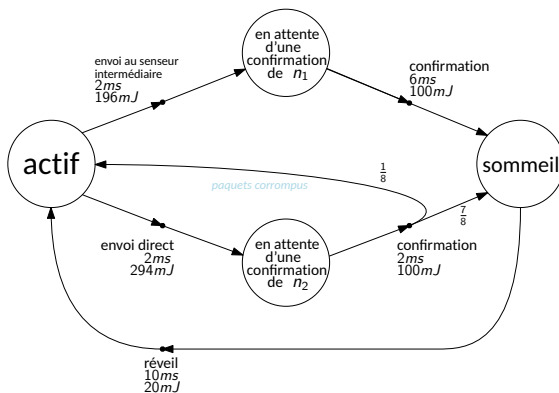
Un MDP à  $d \in \mathbb{N}_0$  dimensions est un tuple  $\mathcal{M} = (S, A, \Delta, w)$  tel que

- $S, A, \Delta$  sont définis de la même façon que pour un MDP classique
- $w : A \rightarrow \mathbb{N}_0^d$  est une fonction de coût, associant chaque action à un coût (strictment positif) par dimension du MDP.

# MDP multidimensionnel

## Exemple

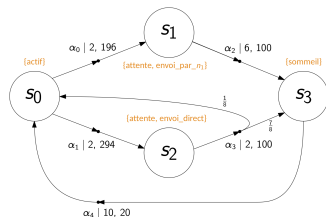
On ajoute le coût en énergie de l'envoi d'un message de  $n_0$  à  $n_2$



# Requêtes percentiles dans les MDP multi-dimensionnels

## Motivation :

- satisfaire plusieurs requêtes d'accessibilité limité par un coût dans un MDP multidimensionnel.

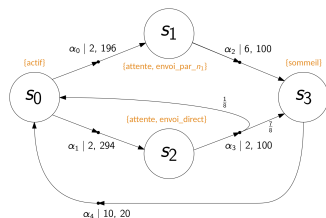


- $Q_1 := \mathbb{P}_{s_0}^{\sigma_1}(\Diamond_1 : \leq 4 \text{ sommeil}) \geq 0.8$
- $Q_2 := \mathbb{P}_{s_0}^{\sigma_2}(\Diamond_2 : \leq 700 \text{ sommeil}) \geq 0.9$
- $\sigma_1$  : envoi direct
- ➡  $\mathbb{P}_{s_0}^{\sigma_1}(\Diamond_1 : \leq 4 \text{ sommeil}) = 0.875$
- $\sigma_2$  : envoi par un noeud intermédiaire
- ➡  $\mathbb{P}_{s_0}^{\sigma_2}(\Diamond_2 : \leq 700 \text{ sommeil}) = 1$

# Requêtes percentiles dans les MDP multi-dimensionnels

## Motivation :

- satisfaire plusieurs requêtes d'accessibilité limité par un coût dans un MDP multidimensionnel.



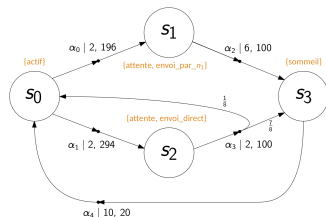
- $Q_1 := \mathbb{P}_{s_0}^{\sigma_1}(\Diamond_1 : \leq 4 \text{ sommeil}) \geq 0.8$
- $Q_2 := \mathbb{P}_{s_0}^{\sigma_2}(\Diamond_2 : \leq 700 \text{ sommeil}) \geq 0.9$
- ↪  $\sigma_1$  : envoi direct  
 ⇒ ne satisfait pas  $Q_2$
- ↪  $\sigma_2$  : envoi par un noeud intermédiaire  
 ⇒ ne satisfait pas  $Q_1$



# Requêtes percentiles dans les MDP multi-dimensionnels

## Motivation :

- satisfaire plusieurs requêtes d'accessibilité limité par un coût dans un MDP multidimensionnel.

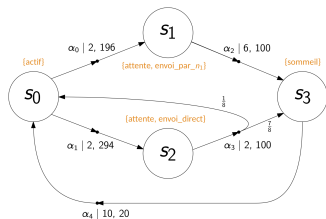
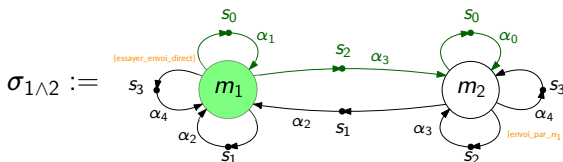


- $Q_1 := \mathbb{P}_{s_0}^{\sigma_1}(\Diamond_{1:\leq 4} \text{sommeil}) \geq 0.8$
- $Q_2 := \mathbb{P}_{s_0}^{\sigma_2}(\Diamond_{2:\leq 700} \text{sommeil}) \geq 0.9$

➡ Résoudre un tel problème requiert une *stratégie à mémoire finie*

# Requêtes percentiles dans les MDP multi-dimensionnels

$\sigma_{1 \wedge 2} :=$  essayer une fois un envoi direct et passer ensuite par  $n_1$  si l'envoi direct a échoué



- $Q_1 := \mathbb{P}_{S_0}^{\sigma_{1 \wedge 2}}(\Diamond_{1:\leq 4} \text{sommeil}) \geq 0.8$
- $Q_2 := \mathbb{P}_{S_0}^{\sigma_{1 \wedge 2}}(\Diamond_{2:\leq 700} \text{sommeil}) \geq 0.9$
- $\mathbb{P}_{S_0}^{\sigma_{1 \wedge 2}}(\Diamond_{1:\leq 4} \text{sommeil}) = 0.875 \models Q_1$
- $\mathbb{P}_{S_0}^{\sigma_{1 \wedge 2}}(\Diamond_{2:\leq 700} \text{sommeil}) = 1 \models Q_2$

$$\rightsquigarrow 394 \leq TS^{\{s_3\}}(\pi) \leq 690$$

$$\forall \pi \in \text{Paths}^{\sigma_{1 \wedge 2}}(s_0)$$

# SSP-PQ

## Definition (SSP-PQ)

Soient  $\mathcal{M} = (S, A, \Delta, w)$ , un MDP multidimensionnel tel que  $w : A \rightarrow \mathbb{N}_0^d$ ,  $s \in S$ , un état de  $\mathcal{M}$  et  $q \in \mathbb{N}$  **contraintes percentiles**.

Ces contraintes percentiles sont décrites par les ensembles d'états cibles  $T_i \subseteq S$ , les **dimensions**  $k_i \in \{1, \dots, d\}$ , les seuils de longueur  $l_i \in \mathbb{N}$  et les seuils de probabilité  $\alpha_i \in [0, 1] \cap \mathbb{Q}$ , pour tout  $i \in \{1, \dots, q\}$ .

Le problème **SSP-PQ** consiste à décider s'il existe une stratégie  $\sigma$  pour laquelle la requête suivante est satisfaite :

$$\mathcal{Q} := \bigwedge_{i \in \{1, \dots, q\}} \mathbb{P}_S^\sigma(\Diamond_{k_i : \leq l_i} T_i) \geq \alpha_i$$

où  $\Diamond_{k_i : \leq l_i} T_i$  dénote l'ensemble des chemins satisfaisant  $\Diamond_{\leq l_i} T_i$  sur la dimension  $k_i$

- peut être décidé en **temps exponentiel**
- **PSPACE-difficile**

# SSP-PQ

## Algorithme

1. Construire  $\mathcal{M}_l$ , le MDP déplié de  $\mathcal{M}$ , de la même façon que pour le problème SSP-P, mais tel que  $l = \max_i l_i$ .
    - $S_l \in S \times (\{0, \dots, l\} \cup \{\perp\})^d$
    - un chemin n'est écarté que lorsque le coût du chemin courant, pour chacune de ses dimensions  $j$ , excède  $\max_i l_i$ , avec  $i \in \{1, \dots, q \mid \mathcal{Q}_i := \mathbb{P}_S^\sigma(\Diamond_{k_i: \leq l_i} T_i) \geq \alpha_i \wedge k_i = j\}$
    - certains chemins peuvent dépasser le seuil de longueur d'une requête tout en restant intéressants pour la stratégie
    - en effet, on ne cherche pas forcément à atteindre un état qui satisfait toutes les contraintes de coûts à la fois !
- ~> notion de **compromis**

# SSP-PQ

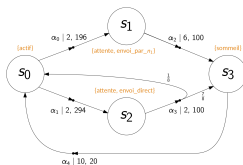
## Algorithme

1. Construire  $\mathcal{M}_I$ , le MDP déplié de  $\mathcal{M}$ , de la même façon que pour le problème SSP-P, mais tel que  $I = \max_i I_i$ .
2. Pour chaque requête  $i \in \{1, \dots, q\}$ , on calcule un ensemble d'états cibles  $R_i$  dans  $\mathcal{M}_I$  pour lequel tous les états contenus dans cet ensemble satisfont la contrainte liée au coût de cette requête.
3. Résoudre le *problème d'accessibilité multiple* aux différent ensembles  $R_i$ , pour chaque requête  $i \in \{1, \dots, q\}$ . On recherche donc une stratégie  $\sigma_i$  qui assure au système d'atteindre les ensembles  $R_i$  depuis  $(s, 0, \dots, 0)$  avec une probabilité  $\alpha_i$ 
  - peut être répondu en temps **polynomial en  $|\mathcal{M}_I|$**  ...
  - ... mais **exponentiel en  $|R_i|$** , i.e.,  $q$

## Algorithme

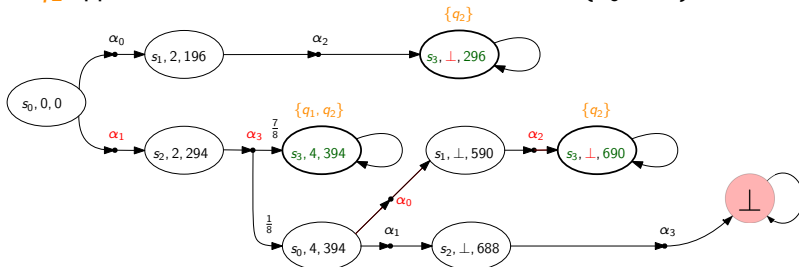
## SSP-PQ

## Algorithme : exemple



- $Q_1 := \mathbb{P}^\sigma_{s_0}(\Diamond_{1:\leq 4} \text{sommeil}) \geq 0.8$
- $Q_2 := \mathbb{P}^\sigma_{s_0}(\Diamond_{2:\leq 700} \text{sommeil}) \geq 0.9$

**Idée clé :** il est nécessaire qu'il y ait une possibilité pour que les labels  $q_1$  et  $q_2$  apparaissent au moins une fois dans le futur de  $(s_0, 0, 0)$  dans  $\mathcal{M}^\sigma$



# SSP-PQ : stratégies

## Definition (Stratégie aléatoire)

Une stratégie aléatoire sans mémoire est une stratégie  $\sigma$  telle que

$$\sigma : S \rightarrow \mathcal{D}(A)$$

où  $\mathcal{D}$  est une distribution de probabilité sur  $A$ . Les stratégies aléatoires à mémoire sont définies d'une façon similaire.

## Théorème

*Les stratégies qui résolvent le problème SSP-PQ ont besoin à la fois de mémoire (sans mémoire dans  $\mathcal{M}_I$ ) et d'aléatoire (pour résoudre le problème d'accessibilité multiple dans  $\mathcal{M}_I$ )*

## SSP-PQ : stratégies

- La construction d'une stratégie requiert de résoudre le problème d'*accessibilité multiple G-SR* dans le MDP déplié jusque  $l$ , i.e.,  $\mathcal{M}_l$
- Le problème d'accessibilité multiple est une généralisation du problème SR dans les MDP.
- Comment construire une telle stratégie ?



# Table des matières

## 1. SP-G

### 1.1 Motivations

### 1.2 Définition

### 1.3 Algorithme

### 1.4 Exemple

## 2. SSP-WE

### 2.1 Motivations

### 2.2 Définition

### 2.3 Algorithme

### 2.4 Exemple

## 3. SSP-PQ

## 3.1 MDP

### multidimensionnels

### 3.2 Motivations

### 3.3 Définition

### 3.4 Algorithme

## 4. G-SR

### 4.1 Définition

### 4.2 Préliminaires

### 4.3 MOLP

### 4.4 Stratégie

### 4.5 Exemple dans storm

# Problème d'accessibilité multiple

## Definition (G-SR)

Soient  $\mathcal{M} = (S, A, \Delta)$ , un MDP,  $s \in S$ , un état de  $\mathcal{M}$  et  $q \in \mathbb{N}$  contraintes.

Ces contraintes sont décrites par les ensembles d'états cibles  $T_i \subseteq S$  et les seuils de probabilité  $\alpha_i \in [0, 1] \cap \mathbb{Q}$ , pour tout  $i \in \{1, \dots, q\}$ .

Le problème G-SR est une généralisation du problème SR qui consiste à décider s'il existe une stratégie  $\sigma$  pour laquelle la requête suivante est satisfaite :

$$\mathcal{Q} := \bigwedge_{i \in \{1, \dots, q\}} \mathbb{P}_s^\sigma(\Diamond T_i) \geq \alpha_i$$

## Nettoyer un MDP

Soient  $\mathcal{M} = (S, A, \Delta)$ , un MDP et  $q \in \mathbb{N}$  contraintes d'accessibilité. On définit l'ensemble  $T$  comme étant l'ensemble des états cibles qui décrivent les  $q$  contraintes, i.e.,  $T = \bigcup_{i \in \{1, \dots, q\}} T_i$ . Nettoyer  $\mathcal{M}$  pour  $T$  consiste à créer un nouveau MDP  $\mathcal{M}_T = (S_T, A_T, \Delta_T)$  tel que

- $A_T = A \cup \{\alpha_T, \alpha_{dead}\}$
- Les états de  $T$  sont absorbants, i.e.,  $\forall t \in T, \Delta_T(t, \alpha_T, t) = 1$  et  $A_T(t) = \{\alpha_T\}$
- $S_T = (S \setminus S_0) \cup \{s_{dead}\}$ , avec  $S_0 = \{s \in S \mid \forall \sigma, \mathbb{P}_s^\sigma(\Diamond T) = 0\}$ 
  - Pour chaque état  $s \in S \setminus T$ , on peut facilement vérifier si  $\exists \sigma$  tel que  $\mathbb{P}_s^\sigma(\Diamond T) > 0$  (si il existe un chemin de  $s$  à  $T$  dans le graphe sous-jacent de  $\mathcal{M}$ )

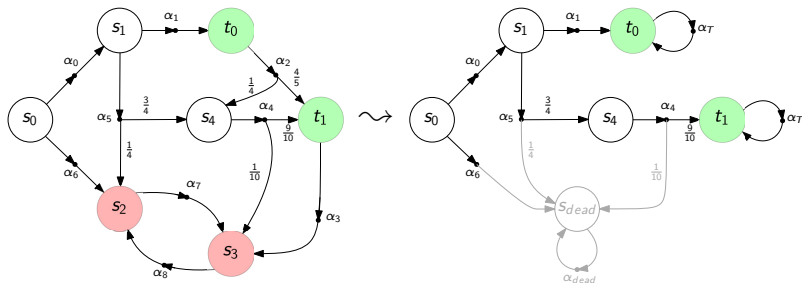
↪ On a retiré les mauvais états de  $S$ , i.e., on crée un état  **$s_{dead}$  invisible absorbant**

$$\Rightarrow \forall s \in S, \sum_{\alpha \in A(s)} \Delta_T(s, \alpha, s') \leq 1$$

↪ **Transition invisible** :  $\Delta_T(s, \alpha, s_{dead}) = 1 - \sum_{s' \in Succ(s, \alpha)} \Delta_T(s, \alpha, s')$

# Nettoyer un MDP

## Exemple



## Courbe de Pareto

- Ensemble des vecteurs  $p$  de points  $p_1, \dots, p_q$  réalisables qui satisfont les contraintes de probabilités  $\alpha_i$ , i.e.,  $p \geq \alpha$  tel qu'il n'existe pas de vecteur  $p'$  qui domine  $p$ , i.e., tel que  $\neg \exists p', p \leq p'$
- $U_Q = \{p \in [0, 1]^q \cap \mathbb{Q} \mid \exists \sigma, t^\sigma \geq p \wedge p \geq \alpha\}$ , avec  $t^\sigma = \mathbb{P}_S^\sigma(\Diamond T_i)$ 
  - Trop coûteux à calculer !
  - Notion de courbe de Pareto
- $\mathcal{P} \subseteq U_Q$ , la *courbe de Pareto de  $U_Q$* , contient tous les vecteurs  $p$  *Pareto optimaux*  $\approx$  *courbe des compromis*.
- $p \in U_Q$  est Pareto-optimal ssi  $\neg \exists p' (p' \in U_Q \wedge p \leq p' \wedge p \neq p')$
- La courbe de Pareto est en général une surface polyédrale de taille superpolynomiale

# Programme Linéaire multi-objectif

On résout le problème G-SR via un *programme linéaire à objectifs multiples*.

- Un programme linéaire à objectifs multiples (MOLP) se définit de la même façon qu'un LP classique à l'exception du fait qu'on cherche à satisfaire **simultanément** plusieurs fonctions objectifs.
- Si les objectifs sont non-triviaux, **il n'existe pas de solution unique qui optimise tous les objectifs simultanément**
  - ↪ compromis
  - ↪ pas de solution optimale
- Une solution acceptable d'un MOLP est une solution *non-dominante*
  - ↪ aucune fonction objectif ne peut être optimisée sans dégrader la valeur d'une autre fonction objectif.
- Soit  $V$ , l'ensemble des solutions qui satisfont toutes les contraintes du MOLP. L'ensemble des solutions acceptables du MOLP est la courbe de pareto  
 $\mathcal{P} \subseteq V$

# Courbe de Pareto

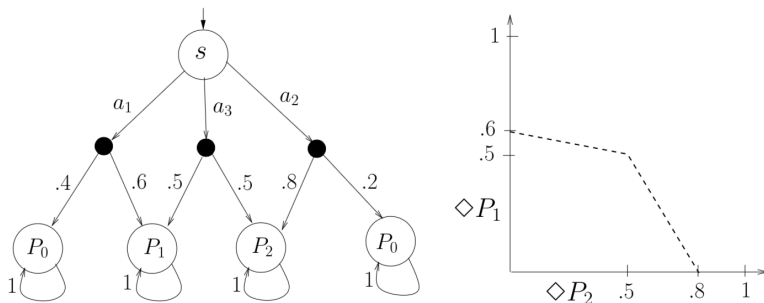


Figure – MDP avec deux objectifs,  $\diamond P_1$  et  $\diamond P_2$ , ainsi que la courbe de Pareto associée [1]

## G-SR

### Programme Linéaire multi-objectif

Supposons que le MDP est nettoyé pour  $T$ . Le MOLP se définit comme suit :

**Objectifs**  $i \in \{1, \dots, q\}$  :  $\max \sum_{t \in T_i} y_t$

sous les contraintes

$$\sum_{s' \in \text{Pred}(t) \setminus T} \sum_{\alpha' \in A(s')} \Delta(s', \alpha', t) \cdot y_{s', \alpha'} = y_t \quad \forall t \in T$$

$$\mathbb{1}(s) + \sum_{s' \in \text{Pred}(s)} \sum_{\alpha' \in A(s')} \Delta(s', \alpha', s) \cdot y_{s', \alpha'} = \sum_{\alpha \in A(s)} y_{s, \alpha} \quad \forall s \in S \setminus T$$

$$y_t \geq 0 \quad \forall t \in T_i$$

$$y_{s, \alpha} \geq 0 \quad \forall s \in S \setminus T \text{ et } \alpha \in A(s)$$

- $\mathbb{1}(s) = 1$  ssi  $s \in S$  est l'état pour lequel on cherche à satisfaire  
 $\bigwedge_{i \in \{1, \dots, q\}} \mathbb{P}_s^\sigma(\Diamond T_i) \geq \alpha_i$
- Ce MOLP est dérivé du LP dual de celui du problème SR pour lequel la solution optimale est  $x_s^* = \max_\sigma \mathbb{P}_s^\sigma(\Diamond T)$



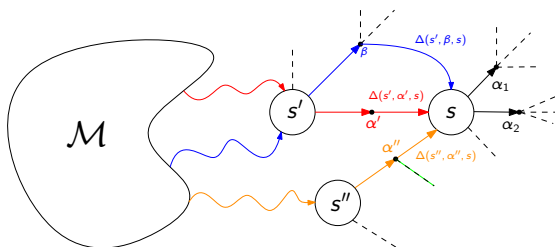
## G-SR

## MOLP

$$\mathbb{1}(s) + \sum_{s' \in \text{Pred}(s)} \sum_{\alpha' \in A(s')} \Delta(s', \alpha', s) \cdot y_{s', \alpha'} = \sum_{\alpha \in A(s)} y_{s, \alpha} \quad \forall s \in S \setminus T$$

$y_{s, \alpha}$  = espérance du nombre de fois que la transition  $s \xrightarrow{\alpha}$  est empreintée

$\sum_{\alpha \in A(s)} y_{s, \alpha}$  = espérance du nombre de fois que l'état  $s$  est visité



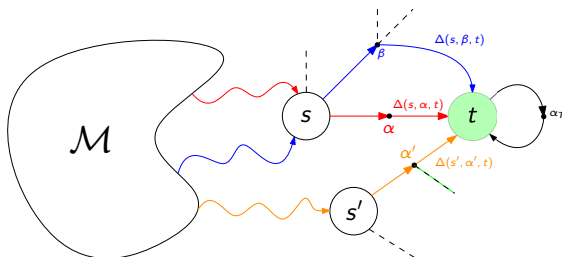
## G-SR

## MOLP

$$y_t = \sum_{s' \in \text{Pred}(t) \setminus T} \sum_{\alpha' \in A(s')} \Delta(s', \alpha', t) \cdot y_{s', \alpha'} \quad \forall t \in T$$

$y_t$  = espérance du nombre de fois qu'une transition atteint  $t$  *pour la première fois*

↪ il s'agit d'une autre façon de formuler la probabilité d'atteindre éventuellement  $t$



G-SR

MOLP

$$\sum_{t \in T_i} y_t$$

correspond donc à la probabilité d'atteindre  $T_i$

↪ Si une solution acceptable du MOLP  $y'$  existe, alors il reste à vérifier que

$$\sum_{t \in T_i} y'_t \geq \alpha_i \quad \forall i \in \{1, \dots, q\}$$

# G-SR

## Théorème

Soient  $\mathcal{M} = (S, A, \Delta)$ , un MDP,  $s \in S$ , un état de  $\mathcal{M}$  et  $q \in \mathbb{N}$  contraintes décrites par les ensembles d'états cibles  $T_i \subseteq S$  et les seuils de probabilités  $\alpha_i \in [0, 1] \cap \mathbb{Q}$ , pour tout  $i \in \{1, \dots, q\}$ . Supposons que  $\mathcal{M}$  est nettoyé pour  $T = \bigcup_{i \in \{1, \dots, q\}} T_i$ . Les propositions suivantes sont équivalentes :

(1.) Il existe une stratégie aléatoire sans mémoire telle que

$$\bigwedge_{i=1}^q \mathbb{P}_s^\sigma(\Diamond T_i) \geq \alpha_i$$

(2.) Il existe une solution acceptable  $y'$  au MOLP telle que

$$\bigwedge_{i=1}^q \sum_{t \in T_i} y'_t \geq \alpha_i$$

## G-SR

### Construction de la stratégie

Supposons que  $y'$  est une solution acceptable du MOLP tel que

$$\sum_{t \in T_i} y'_t \geq \alpha_i > 0 \quad \forall i \in \{1, \dots, q\} \quad (1)$$

Soit  $V = \{s \in S \setminus T \mid \sum_{\alpha \in A(s)} y'_{s,\alpha} y'_{s,\alpha} > 0\}$ , les états par lesquels on passe afin de satisfaire (1), alors  $\forall s_V \in V, \alpha \in A(s_V)$

$$\begin{aligned} \sigma(s_V)(\alpha) &:= \frac{y'_{s_V,\alpha}}{\sum_{\alpha' \in A(s_V)} y'_{s_V,\alpha'}} \\ &:= \frac{\mathbb{E}_S(\# \text{ transition } s_V \xrightarrow{\alpha} \text{ est empreintée})}{\mathbb{E}_S(\# \text{ visite de l'état } s_V)} \end{aligned}$$

Note : Vu que  $\sum_{\alpha \in A(s_V)} y'_{s_V,\alpha} > 0$ ,  $\sigma(s_V)$  est une distribution de probabilité sur  $A(s_V)$ , i.e.,  $\mathcal{D}(A(s_V))$ . Pour les autres états  $s' \notin V$ ,  $\sigma(s')$  est une distribution de probabilité aléatoire sur  $A(s')$ , i.e.,  $\mathcal{D}(A(s'))$

# G-SR

## Stratégie et problème de flots

### Lemme

La stratégie  $\sigma$ , construite de cette manière, satisfait  $\bigwedge_{i=1}^q \mathbb{P}_S^\sigma(\Diamond T_i) \geq \alpha_i$

**Idée :**  $y'$ , une solution acceptable du MOLP, définit un **flot stochastique** dont la **source** est  $S$  et dont les **puits** sont les états de  $T$ .

- Par conservation de flots, les états de  $s \in S \setminus T$  qui ont un nombre de flots sortants positifs ( $\#_{\alpha \in A(s)} s \xrightarrow{\alpha}$ ) — et donc un nombre de flots entrants positifs
  - doivent être tous accessibles depuis la source
  - doivent tous atteindre  $T$
  - ne peuvent pas accéder aux états avec un flot nul

## G-SR

### *Stratégie et problème de flots*

- $\sigma$  est obtenue en normalisant les flots sortant de chaque action vers les états avec un nombre de flots sortants positif.
- En utilisant  $\sigma$  définie de cette façon, l'espérance du nombre de fois que  $\alpha \in A(s)$  est choisie alors que le système se trouve en l'état  $s$  est donné par  $y'_{s,\alpha}$ .
- Par conséquent, vu que les transitions vers un état  $t \in T$  depuis un état de  $S \setminus T$  sont employées pour atteindre  $t$  une **unique fois**, alors, les contraintes définissant  $y'_t$  induisent  $y'_t = \mathbb{P}_s^\sigma(\Diamond\{t\})$

Exemple dans storm

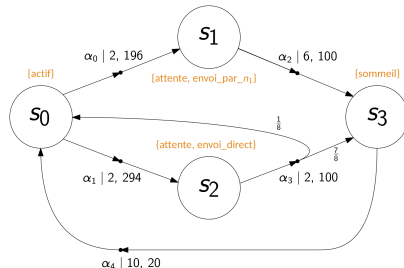
# SSP-PQ et GSR

## Exemple dans Storm

```

1  mdp
2
3  module sensors_transmission
4
5  s: [0..3] init 0;
6
7  [alpha0] s=0 -> 1 : (s'=1);
8  [alpha1] s=0 -> 1 : (s'=2);
9  [alpha2] s=1 -> 1 : (s'=3);
10 [alpha3] s=2 -> 0.125 : (s'=0) + 0.875 : (s'=3);
11 [alpha4] s=3 -> 1 : (s'=0);
12
13 endmodule
14
15 label "sleep" = s=3;
16
17 rewards "time"
18   [alpha0] true : 2;
19   [alpha1] true : 2;
20   [alpha2] true : 6;
21   [alpha3] true : 2;
22   [alpha4] true : 10;
23 endrewards
24
25 rewards "energy"
26   [alpha0] true : 196;
27   [alpha1] true : 294;
28   [alpha2] true : 100;
29   [alpha3] true : 100;
30   [alpha4] true : 20;
31 endrewards

```





Exemple dans storm

## SSP-PQ et GSR

### Exemple dans Storm

$$\begin{aligned} & \stackrel{?}{\exists \sigma, \mathbb{P}_{s_0}^\sigma (\Diamond_{1:\leq 4} \text{sommeil}) \geq 0.8 \wedge \mathbb{P}_{s_0}^\sigma (\Diamond_{2:\leq 700} \text{sommeil}) \geq 0.9} \\ & \equiv_{\text{PRCTL}} s_0 \stackrel{?}{\models} \text{multi}(\mathcal{P}_{\geq 0.8}^{\max}(\Diamond_{1:\leq 4} \text{sommeil}), \mathcal{P}_{\geq 0.9}^{\max}(\Diamond_{2:\leq 700} \text{sommeil})) \end{aligned}$$

```
storm --prism ../../models/sensors.prism --prop "multi(Pmax>=0.8 [F{\"time\"}<=4 \"sleep\"], Pmax>=0.9 [F{\"energy\"}<=700 \"sleep\"])"
Storm 1.2.0
```

```
Command line arguments: --prism ../../models/sensors.prism
                        --prop multi(Pmax>=0.8 [F{"time"}<=4 "sleep"], Pmax>=0.9 [F{"energy"}<=700 "sleep"])
```

Time for model construction: 0.302s.

-----

```
Model checking property multi(Pmax>=4/5 [true Urew{"time"}<=4 "sleep"], Pmax>=9/10 [true Urew{"energy"}<=700 "sleep"]) ...
Result (for initial states): true
```

Time for model checking: 0.003s.

Exemple dans storm

## SSP-PQ et GSR

### Exemple dans Storm

$$\max_{\sigma} (\mathbb{P}_{s_0}^{\sigma}(\Diamond_{1:\leq 4} \text{sommeil}), \mathbb{P}_{s_0}^{\sigma}(\Diamond_{2:\leq 700} \text{sommeil}))$$

$$\equiv_{\text{PRCTL}} s_0 \models \text{multi}(\mathcal{P}_{=?}^{\max}(\Diamond_{1:\leq 4} \text{sommeil}), \mathcal{P}_{=?}^{\max}(\Diamond_{2:\leq 700} \text{sommeil}))$$

```
storm --prism ../../models/sensors.prism --prop "multi(Pmax=? [F{"time"}<=4 "sleep"], Pmax=? [F{"energy"}<=700 "sleep"])"
Storm 1.2.0
```

```
Command line arguments: --prism ../../models/sensors.prism
                        --prop multi(Pmax=? [F{"time"}<=4 "sleep"], Pmax=? [F{"energy"}<=700 "sleep"])
```

Time for model construction: 0.452s.

Model checking property multi(Pmax=? [true Urew{"time"}<=4 "sleep"], Pmax=? [true Urew{"energy"}<=700 "sleep"]) ...

Result (for initial states):

Underapproximation of achievable values: Polytope with 2 Halfspaces:

```
(      1,      0) * x <= 0.875
(      0,      1) * x <= 1
```

Overapproximation of achievable values: Polytope with 2 Halfspaces:

```
(      1,      0) * x <= 0.875
(      0,      1) * x <= 1
```

1 pareto optimal points found

(Note that these points are safe, i.e., contained in the underapproximation, but there is no guarantee for optimality):

```
(      0.875,      1 )
```

Time for model checking: 0.040s.

## Références I

- [1] Kousha Etessami et al. “Multi-Objective Model Checking of Markov Decision Processes”. In : *Logical Methods in Computer Science* 4.4 (2008). doi : [10.2168/LMCS-4\(4:8\)2008](https://doi.org/10.2168/LMCS-4(4:8)2008).
- [2] Mickael Randour, Jean-François Raskin et Ocan Sankur. “Variations on the Stochastic Shortest Path Problem”. In : *Verification, Model Checking, and Abstract Interpretation - 16th International Conference, VMCAI 2015, Mumbai, India, January 12-14, 2015. Proceedings*. 2015, p. 1-18. doi : [10.1007/978-3-662-46081-8\\_1](https://doi.org/10.1007/978-3-662-46081-8_1).