# Markov döntési folyamatok verifikációja

**Tulics Miklós Gábriel**
**Neptun-kód: VT63UO**

# Markov döntési folyamatok verifikációja


- **Smart Sampling for Lightweight Verification of Markov Decision Processes**

- Pedro D'Argenio,
- Axel Legay †,
- Sean Sedwards †,
- Louis-Marie Traonouez †.

- Universidad Nacional de C´ordoba, Argentina
- † Inria Rennes – Bretagne Atlantique, France

# Pedro D’Argenio

- Tudományos munkatárs a CONICET-nél (The National Scientific and Technical Research Council (Spanish: Consejo Nacional de Investigaciones Científicas y Técnicas, CONICET))
- Egyetemi docens a Universidad Nacional de Cordoba-nál
- Szoftverfejlesztést tanít

- Rare Event Simulation with Fully Automated Importance Splitting. EPEW 2015: 275-290
- SOS rule formats for convex and abstract probabilistic bisimulations. EXPRESS/SOS 2015: 31-45

# Absztrakt

- Markov döntési folyamatokat előnyösek konkruens rendszerek modelljeinek optimalizálásában.
- Markov decision processes (MDP)
- Hogy a Markov döntési folyamatok ellenőrzését el lehessen végezni Monte-Carlo módszerrel szükség van egy ütemezőre, a nem-determinista jelleg miatt.
- Lightweight technikák segítségével a mintákat közvetlenül az ütemező teréből érhetjük el, optimális ütemező találása viszont problematikus.
- A cikk egy „okos” mintavételi algoritmust mutat be.

# Bevezetés

- Markov döntési folyamatok olyan rendszereket írnak le, amelyekben nemdeterminisztikus jelenségek és valószínűségi átmenetek jellemeznek, így nagyon sok valós problémát le tud írni.
- A rendszer állhat több valószínűségi alhálózatból, ahol az átmenetek a többi alhálózat állapotától függenek.

# A Monte-Carlo-módszerről röviden

- Sztochasztikus szimulációs módszer
- Előállítja egy adott kísérlet végeredményét, ezek után az eredményként kapott numerikus jellemzőket feljegyzik és kiértékelik
- Az eredmény hibájának meghatározása szórás kiszámításával történik
- Hasonló véletlenszámokat lehetne generálni a kaszinók kedvelt játékával, a rulettel is.
- Ezért nevezte el a módszer kifejlesztője, Neumann János „Monte-Carlo”-módszernek.

# Általános lépések és a π meghatározása

1. Határozzuk meg egy probléma lehetséges bemeneteit
2. Hozzunk létre véletlenszerű bemeneteket a problémára specifikus valószínűségi eloszlással
3. Végezzünk determinisztikus számításokat a bemeneteken
4. Összesítsük az eredményeket

- Adott egy egységnyi területre beírt körünk. Tudva, hogy a kör és a négyzet területének az aránya π/4, π meghatározható:

1. Rajzoljunk egy négyzetet a földre , majd helyezzünk el egy kört benne
2. Egyenletesen szórjunk szét egységnyi méretű objektumokat (homokszemek, rizsszemek)
3. Számoljuk meg az objektumokat a körön belül és azon kívül
4. A két érték aránya arányos a két területtel, ami π/4, π meghatározható.

# Diszkrét Markov láncok

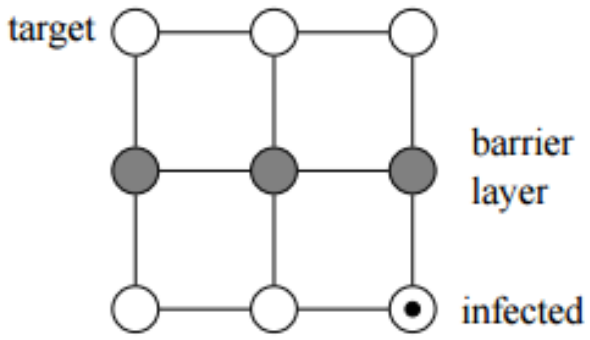



Figure 1: Model of network virus infection.

- Kiszámolhatjuk, hogy mennyi a valószínűsége, hogy a cél csomópont fertőzött lesz.
- Ha tudjuk az egyik csomópontról a másikra való fertőzés valószínűségét, akkor a folyamatot egy diszkrét Markov lánccal modellezhetjük.
- Viszont lehet, hogy a vírus megválasztja, hogy melyik csomópontot fertőzi meg (nagyobb esélye legyen a zárórétegen átjutnia). -> néhány csomópont biztosan meg fog fertőződni, a többire csekélyebb az esély.
- A vírus választását egy nemdeterminisztikus átmenettel modellezhetjük. A maximális és minimális valószínűségeket meg lehet határozni.

# Markov döntési folyamat egy szakasza

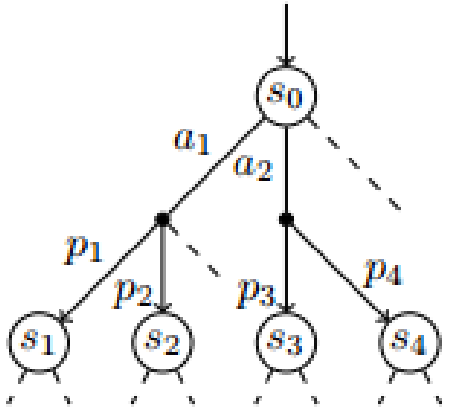



Figure 2: Fragment of a Markov decision process.

- s = state
- a = action
- p = valószínűségi átmenetek (nemdeterminisztikus)
- Az ütemező (scheduler) a cikkben a nemdeterminizmus megoldására utal egy Markov döntési folyamat
- Egy ütemező kiválasztja a „futásra jelentkező” feladatok közül a futókat. Regisztrál, feldolgozza a jelentkezéseke (pl. prioritási szinteket rendel hozzá), a kiválasztott feladatokat szálakhoz rendeli hozzá.
- Ebben a munkában az MDP-ket valószínűségi rendszerek modellellenőrzésre használják, a cél, hogy megtalálják azokat az ütemezőket, amelyek maximalizálják vagy minimalizálják a rendszer egy tulajdonságának a valószínűségét.

# Fontos fogalmak

- Statisztikai modell ellenőrzés (Statistical model checking (SMC)) Monte Carlo mintavételi technikákat gyűjteménye, statisztikai megbízhatósági mértékeket rendel a problémákhoz.
- "Lightweight", azaz memória-hatékony MDP ellenőrzés. Megnyitja az utat, hogy megoszthatjuk magas teljesítményű párhuzamos architektúrák között, mint amilyen a grafikai processzor (GPGPU).
- „Okos mintavételezési" („smart sampling") algoritmusok jelentősen jobb szimulációs költségvetéssel rendelkeznek. Egy adott számú jelölt ütemezőre, az okos mintavétel csökkentheti a szimuláció költségét a valószínűség-becslésnek N/[2+log2N] -el, ahol az N a minimális szimuláció szám, ahhoz, hogy statisztikailag megbízható eredményt kapjunk.
- Implementálják a mintavételi eljárást PLASMA (Platform for Learning and Advanced Statistical Model checking Algorithms) platformban és esettanulmányokban mutatják be a sikerességét.

# Smart Estimation algorithm

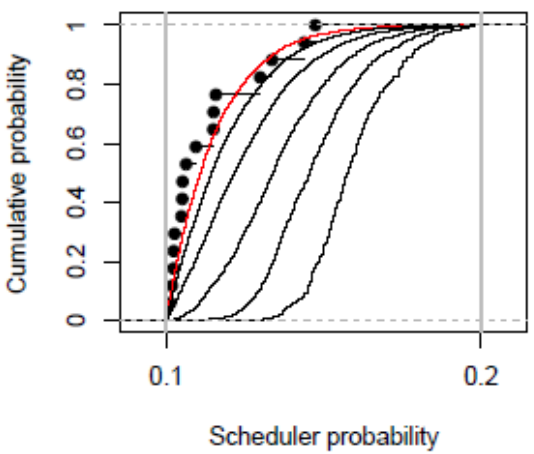
**Algorithm 4:** Smart Estimating

**Input**:
$\mathcal{M}$: an MDP
$\varphi$: a property
$\epsilon, \delta$: the required Chernoff bound
$N_{\max} > \ln(2/\delta)/(2\epsilon^2)$: the per-iteration budget
**Output**: $\hat{p}_{\overline{\max}} \approx p_{\max}$, where $p_{\overline{\max}} \approx p_{\max}$ and $\mathrm{P}(|p_{\overline{\max}} - \hat{p}_{\overline{\max}}| \geq \epsilon) \leq \delta$

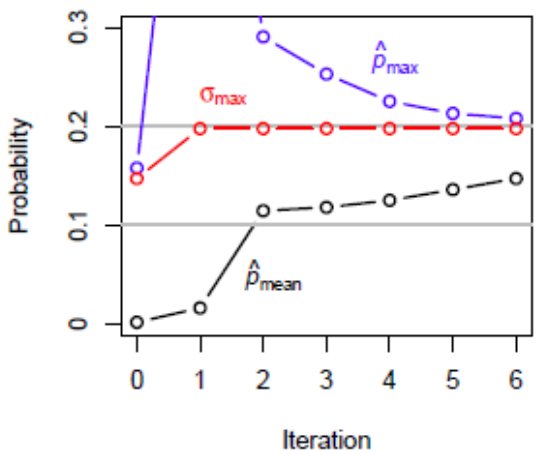
$N \leftarrow \lceil \sqrt{N_{\max}} \rceil$; $M \leftarrow \lceil \sqrt{N_{\max}} \rceil$
$S \leftarrow \{M$ seeds chosen uniformly at random$\}$
$\forall \sigma \in S, \forall i \in \{1, \ldots, N\} : \omega_i^\sigma \leftarrow \mathrm{Simulate}(\mathcal{M}, \varphi, \sigma)$
$R : S \to \mathbb{N}$ maps scheduler seeds to number of traces satisfying $\varphi$:
$R \leftarrow \{(\sigma, n) \mid \sigma \in S \wedge \mathbb{N} \ni n = \sum_{i=1}^{N} \mathbf{1}(\omega_i^\sigma \models \varphi)\}$
$\hat{p}_{\overline{\max}} \leftarrow \max_{\sigma \in S}(R(\sigma)/N)$
$N \leftarrow \lceil 1/\hat{p}_{\overline{\max}} \rceil$, $M \leftarrow \lceil N_{\max} \hat{p}_{\overline{\max}} \rceil$
$S \leftarrow \{M$ seeds chosen uniformly at random$\}$
$\forall \sigma \in S, \forall i \in \{1, \ldots, N\} : \omega_i^\sigma \leftarrow \mathrm{Simulate}(\mathcal{M}, \varphi, \sigma)$
$R \leftarrow \{(\sigma, n) \mid \sigma \in S \wedge \mathbb{N} \ni n = \sum_{i=1}^{N} \mathbf{1}(\omega_i^\sigma \models \varphi)\}$
$S \leftarrow \{\sigma \in S \mid R(\sigma) > 0\}$
$\forall \sigma \in S,\ R(\sigma) \leftarrow 0$; $i \leftarrow 0$; $conf \leftarrow 1$
**while** $conf > \delta \wedge S \neq \emptyset$ **do**
  $i \leftarrow i + 1$
  $M_i \leftarrow |S|$
  $N_i \leftarrow 0$
  **while** $conf > \delta \wedge N_i < \lceil N_{\max}/M_i \rceil$ **do**
    $N_i \leftarrow N_i + 1$
    $conf \leftarrow 1 - (1 - e^{-2\epsilon^2 N_i})^{M_i}$
    $\forall \sigma \in S : \omega_{N_i}^\sigma \leftarrow \mathrm{Simulate}(\mathcal{M}, \varphi, \sigma)$
  $R \leftarrow \{(\sigma, n) \mid \sigma \in S \wedge \mathbb{N} \ni n = \sum_{j=1}^{N_i} \mathbf{1}(\omega_j^\sigma \models \varphi)\}$
  $\hat{p}_{\overline{\max}} \leftarrow \max_{\sigma \in S}(R(\sigma)/N_i)$
  $R' : \{1, \ldots, |S|\} \to S$ is an injective function s.t. $\forall (n, \sigma), (n', \sigma') \in R',\ n > n' \implies R(\sigma) \geq R(\sigma')$
  $S \leftarrow \{\sigma \in S \mid \sigma = R'(n) \wedge n \in \{\lfloor |S|/2 \rfloor, \ldots, |S|\}\}$

- Olyan ütemezőket keres, ahol egy tulajdonság maximális valószínűségű
- Pmax (a feltételezett/fiktív igaz max valószínűség),
- Pmax_ (a legjobb jelölt valódi valószínűsége)
- Pˆmax_ (a legjobb jelölt ütemező becsült valószínűsége)
- Hipotézis: Létezik ϕ tulajdonság, aminek a valószínűsége θ
- Az algoritmus minden iterációban ad egy becslést ütemezőre, magas konfidenciaszinttel.

# Start Estimation algorithm




(a) Scheduler distributions. Dots denote the results of initial sampling. The red line is the set of schedulers to refine. Black lines show the result of subsequent refinements.




(b) Estimates and schedulers. At each iterative step: $\hat{p}_{\max}$ is the maximum estimate, $\hat{p}_{\text{mean}}$ is the mean estimate and $\sigma_{\max}$ is the true maximum probability of the available schedulers.

Figure 9: Convergence of Algorithm 4 with exponentially distributed scheduler probabilities

- Módosítással az algoritmus hipotézisvizsgálatra is képes.

# Esettanulmányok

- *IEEE 802.11 Wireless LAN Protocol*
- *IEEE 802.3 CSMA/CD Protocol*
- *Choice Coordination*
- *Network Virus Infection*
- *Gossip Protocol*

- Ezek a standard modellek a numerikus modellellenőrzési szakirodalomból valók, többségük megtalálható a PRISM weboldalon.

# Hálózati vírusfertőzés

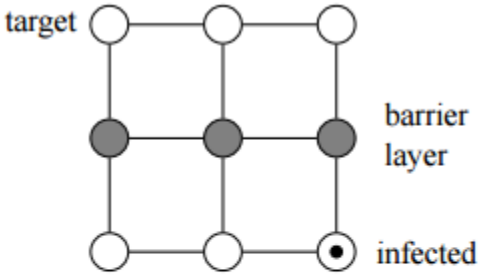



Figure 1: Model of network virus infection.

- A témának növekvő jelentősége van.
- Három csoport csomópont:
  1. egy sor csomópont, amelyek közül egyet egy vírus fertőzött meg (alsó)
  2. Egy sor csomópont, amelyek „tiszták”
  3. És a sor csomópont amely elválasztja a két csoportot.

# Hálózati vírusfertőzés

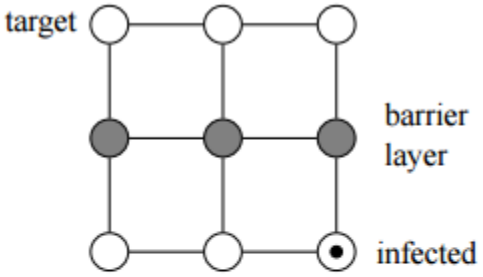
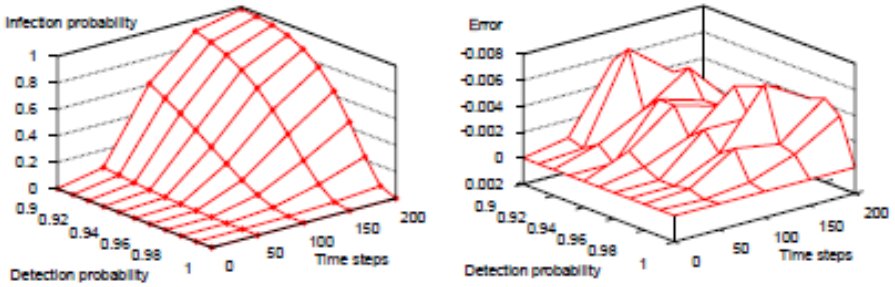



Figure 1: Model of network virus infection.

- A vírus nemdeterminisztikusan kiválasztja mely csomópontot fertőzi meg.
- Egy csomópont valamilyen valószínűséggel észreveszi a vírust és lehetőség van ezt a valószínűséget variálni a köztes rétegben, változtatható paraméterként.
- Az idő egy újabb paraméter.

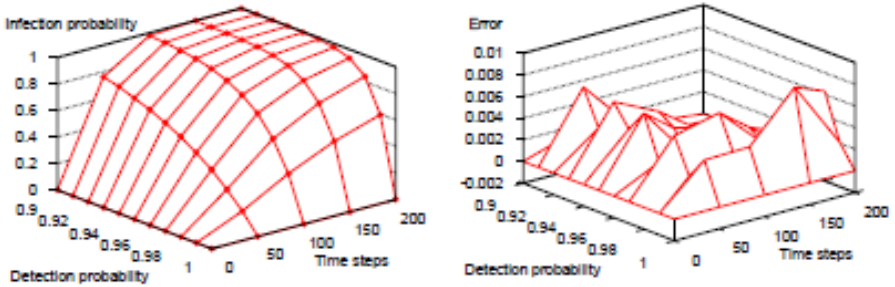
# A hálózati fertőzés minimum és maximum valószínűsége




(a) Estimates. (b) Errors.

Figure 6: Minimum probability of network infection.

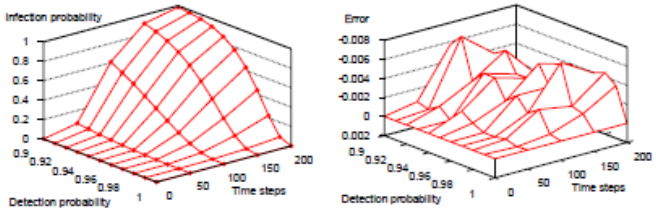
-> A cél csomópont fertőzött lesz.




(a) Estimates. (b) Errors.

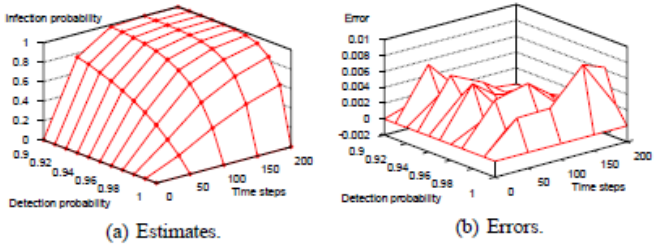
Figure 7: Maximum probability of network infection.

# A hálózati fertőzés minimum és maximum valószínűsége




(a) Estimates. (b) Errors.

Figure 6: Minimum probability of network infection.




(a) Estimates. (b) Errors.

Figure 7: Maximum probability of network infection.

- Becsült minimumok [-0.0070, + 0.00012]
- Becsült maximumok [-0.00012, + 0.0083]

- A negatív értékek a smart algoritmus azon tulajdonságából ered, hogy a becslést kétoldali konvergálással határozza meg.
- Egy pont generálása körülbelül 100 másodpercbe tellett a 6-os ábrán. 64 szimuláció.
- 70 másodpercbe átlagosan a 7-es ábrán.

# Kilátások

- Az eljárás nemdeterminisztikus rendszerekre működik, de használható lehet jutalom alapú Markov döntési folyamatoknál is (*reward-based MDP*)
- A jövőben
  - csökkenteni akarják az ütemezők terét, hogy csak olyan ütemezőkre, amelyek kielégítik az adott tulajdonságot, valamint
  - Szakaszonkénti ütemezőket kívánnak implementálni.
- Ez megakadályozhatja a pontenciális függvényütközéseket.

Köszönöm a figyelmet!