# Optimal Route Synthesis in Space DTN using Markov Decision Processes

Pedro R. D'Argenio

Universidad Nacional de Córdoba – CONICET

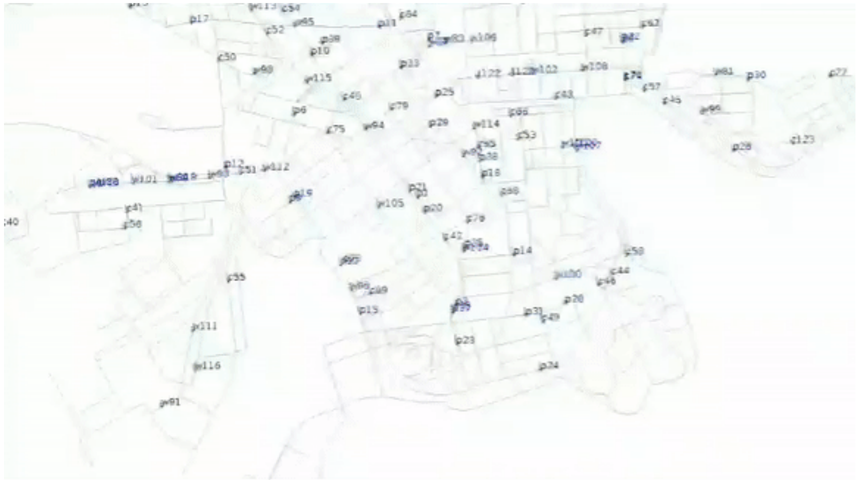
Joint work with
Juan Fraire, Arnd Hartmanns, Fernando Raverta,
Ramiro Demasi, Pablo Madhoery, Jorge Finochietto

ICTAC 2023 - Lima

CONICET

UNC

# Delay Tolerant Networks



- Time-evolving networks lacking continuous and instantaneous end-to-end connectivity
- Routing through "store, carry, and forward" policy
- Contacts can be:
  - Opportunistic: no assumptions can be made on future contacts
  - Predicted: contact patterns can be inferred from history
  - Scheduled: time and duration of contacts can be accurately determined

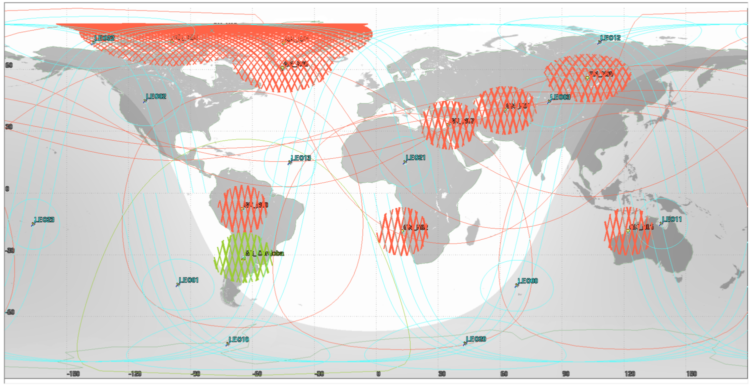
CONICET

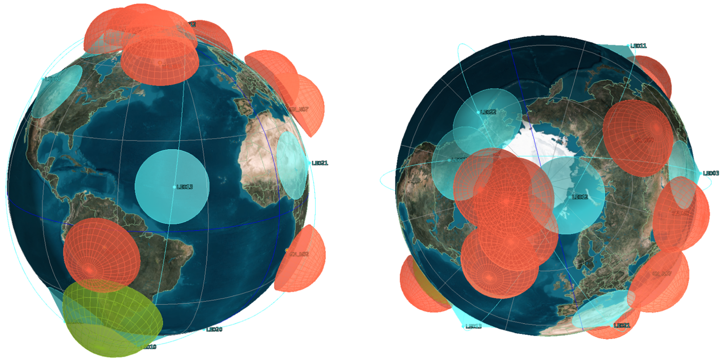
GIF from https://amittharani.com/prophet.html

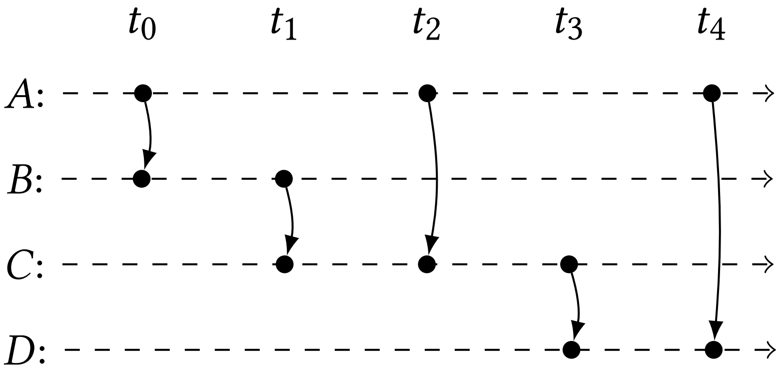
UNC

# Satellite Delay Tolerant Networks

Contact Plan








Standard: Contact Graph Routing (CGR)

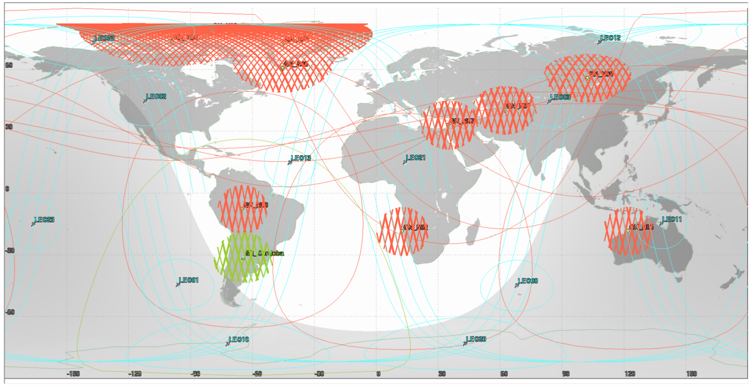
CONICET

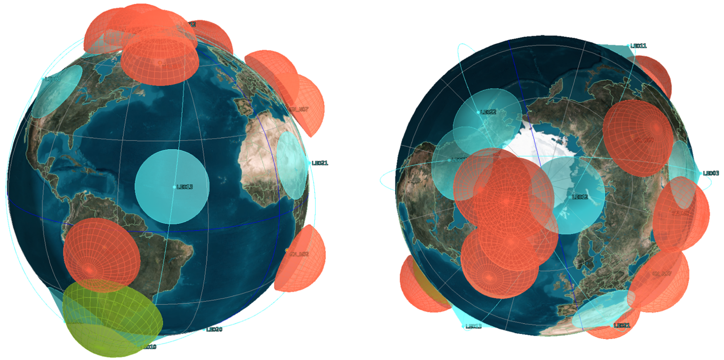
UNC

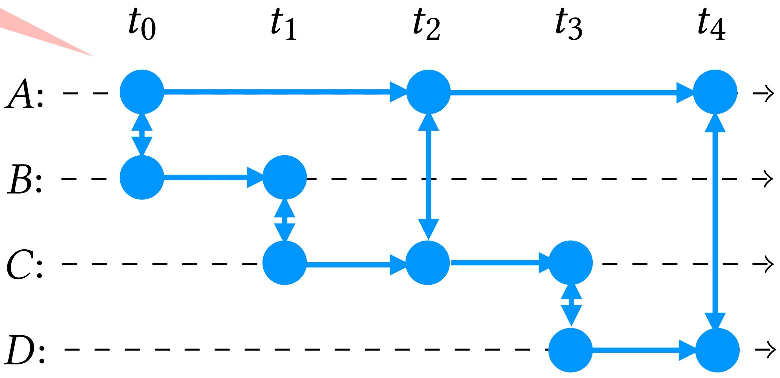
Translates the contact plan to a graph and adapts Dijkstra's algorithm to time dynamics

# lerant Networks

Contact Plan








Standard: Contact Graph Routing (CGR)

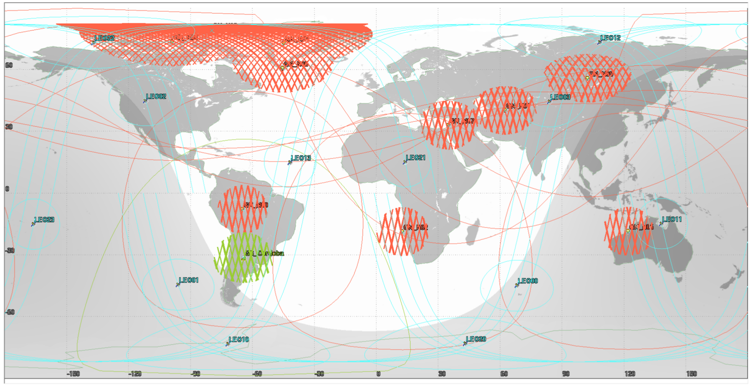
CONICET

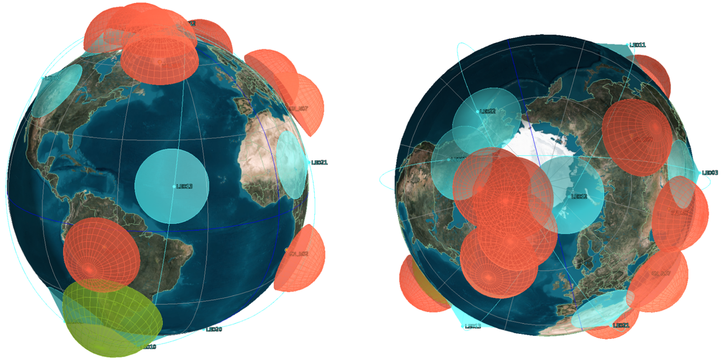
UNC

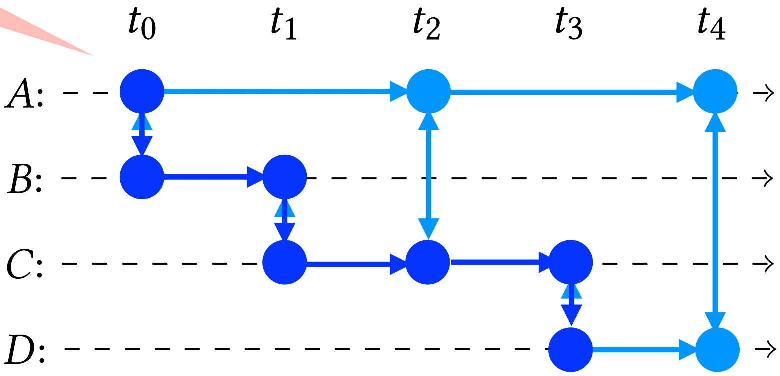
Translates the contact plan to a graph and adapts Dijkstra's algorithm to time dynamics

# lerant Networks

Contact Plan







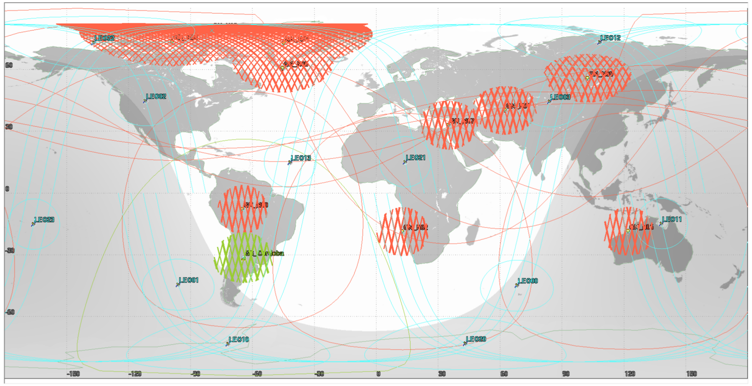

Standard: Contact Graph Routing (CGR)

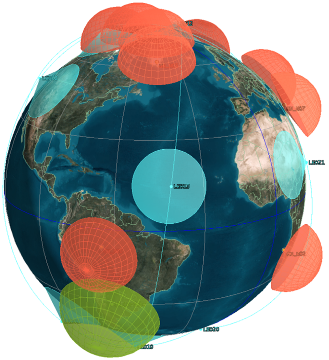
CONICET

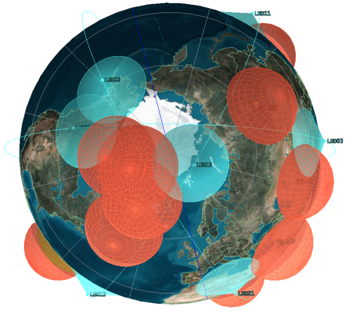
UNC

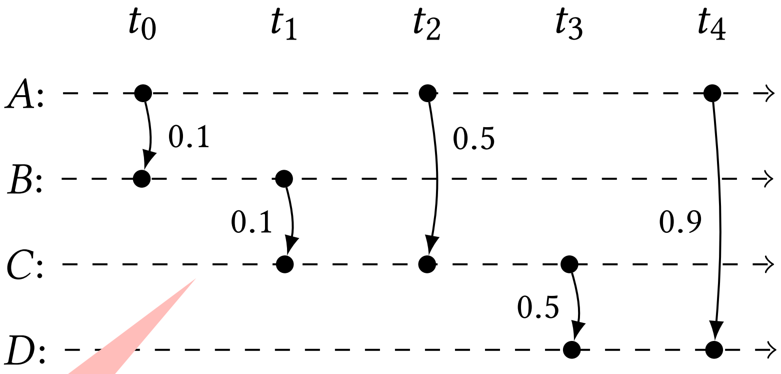
# Satellite Delay Tolerant Networks

Contact Plan *with uncertainties*










Links may fail!

Standard: Contact Graph Routing (CGR)

Increase reliability: CGR with multiple copies

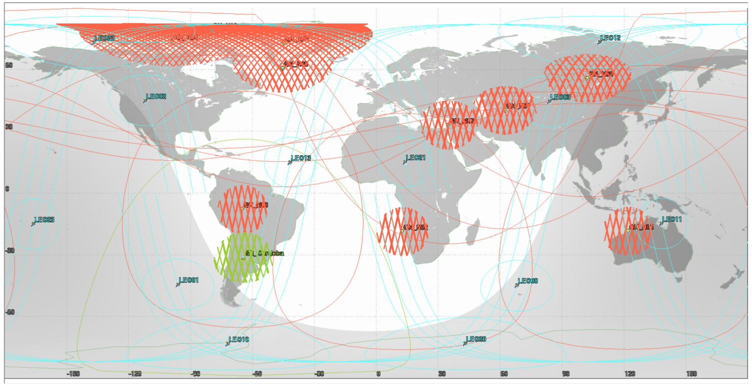
CONICET

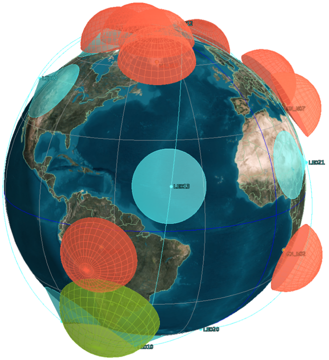
UNC

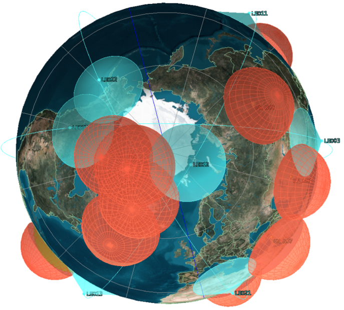
# Satellite Delay Tolerant Networks

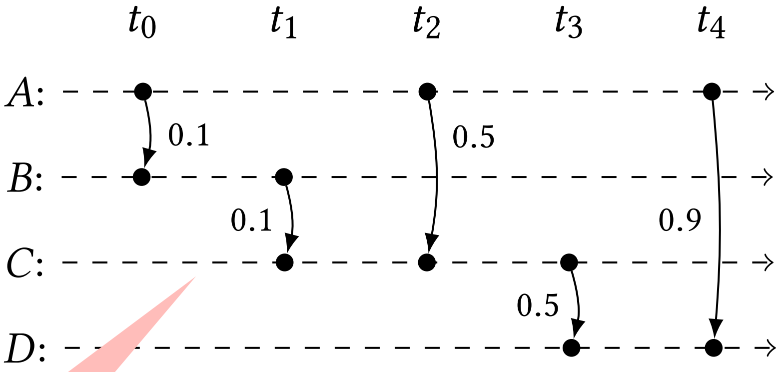
Contact Plan

with uncertainties









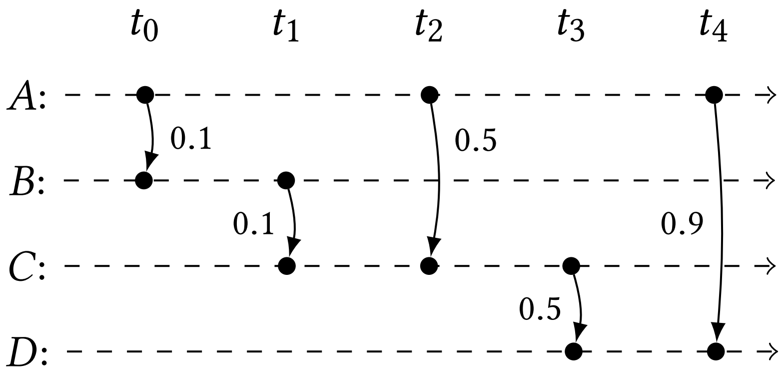

Links may fail!

Not optimal!

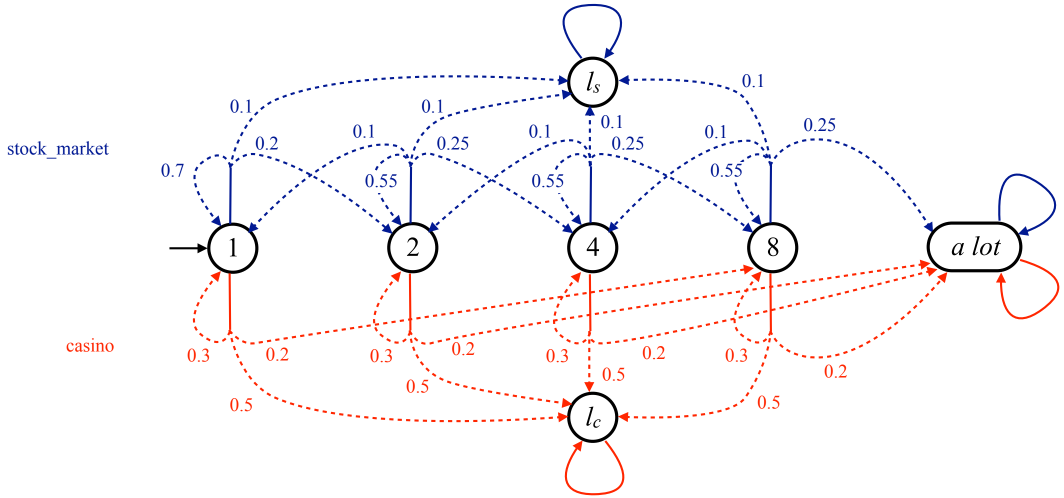
Standard: Contact Graph Routing (CGR)

Increase reliability: CGR with multiple copies

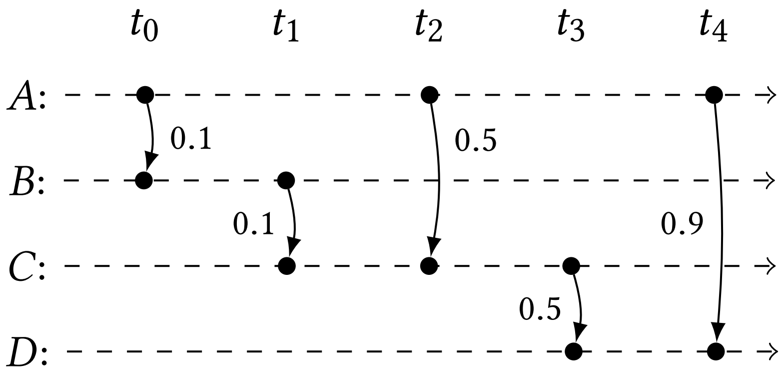
# Optimality through Markov Decision Processes



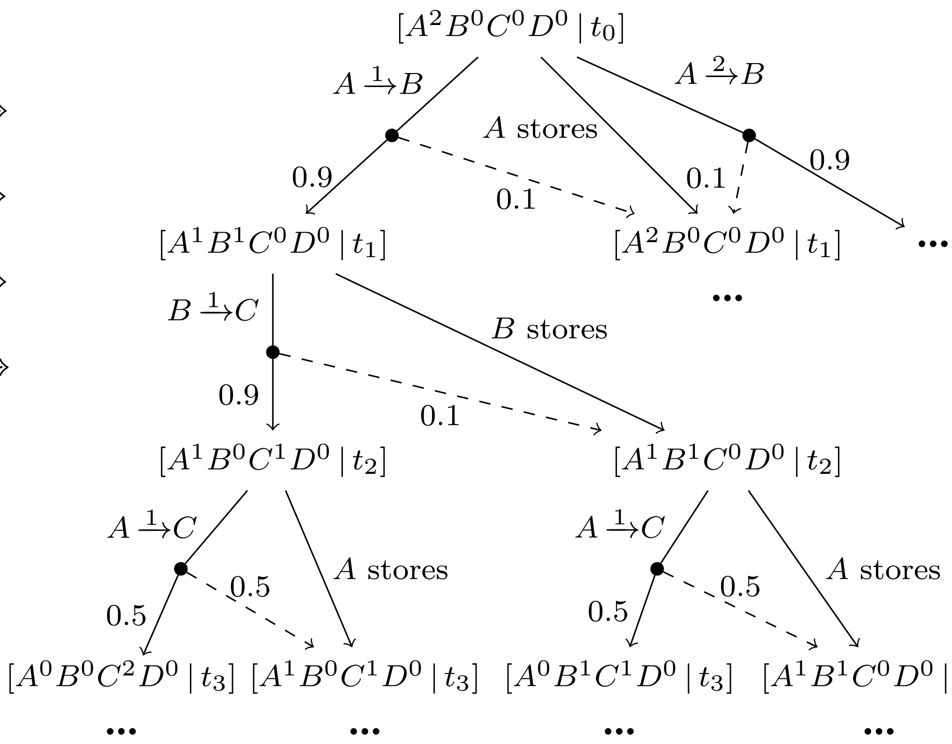

## What is a MDP? (example)



# Optimality through Markov Decision Processes



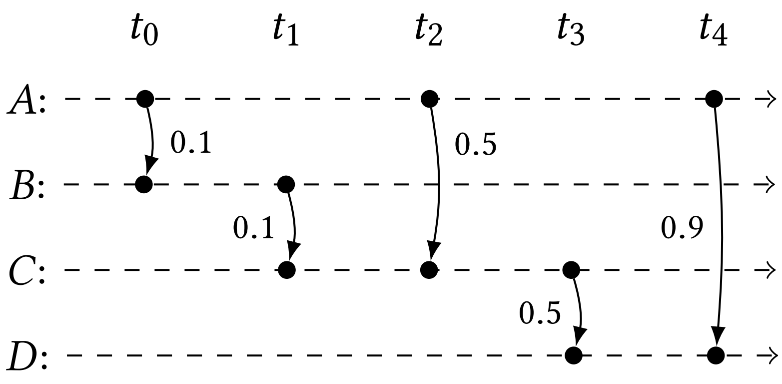

Assume 2 copies are sent

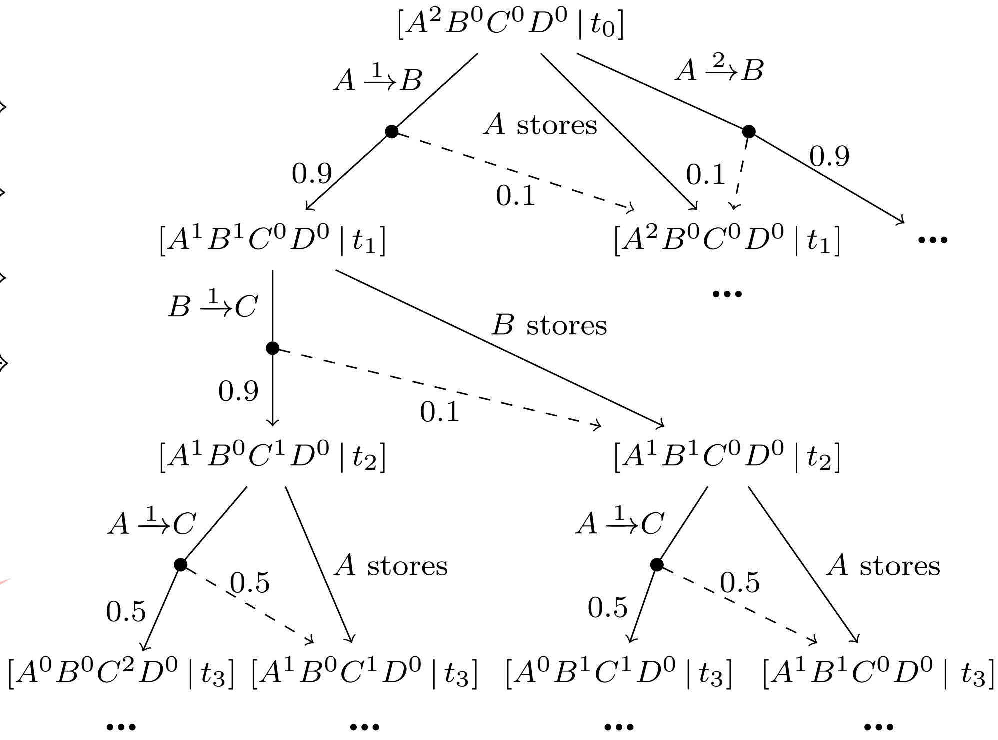



CONICET

UNC

# Optimality through Markov Decision Processes

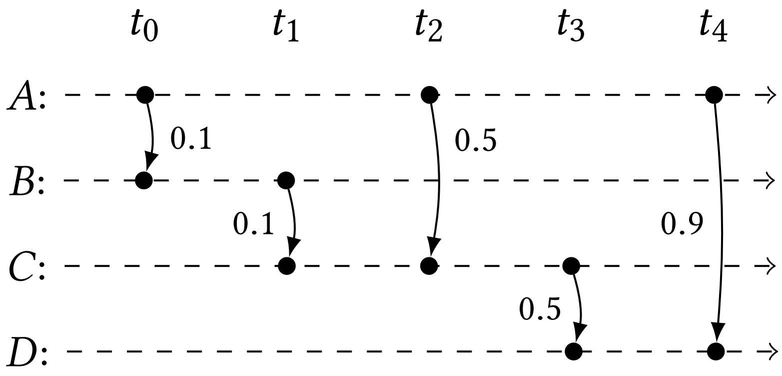



Assume 2 copies are sent

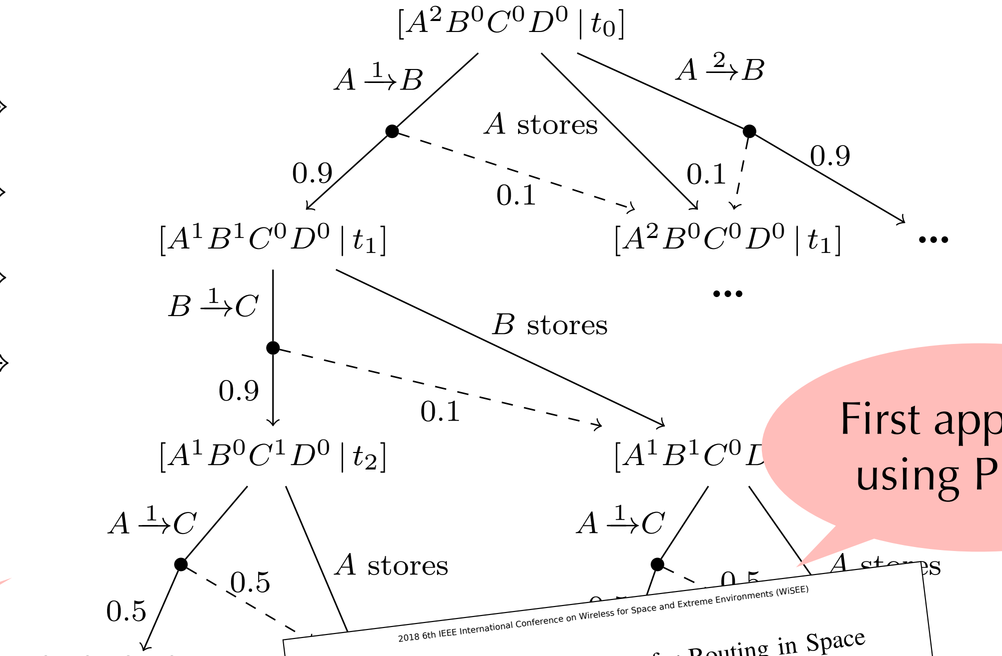
We have a reachability problem where goal states are those with a copy at target node

# Optimality through Markov Decision Processes

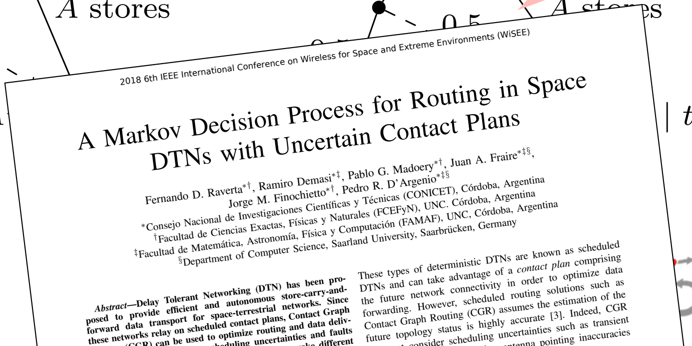



Assume 2 copies are sent

We have a reachability problem where goal states are those with a copy at target node




First approach using PRISM


2018 6th IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)

A Markov Decision Process for Routing in Space DTNs with Uncertain Contact Plans

Fernando D. Raverta*†, Ramiro Demasi*‡, Pablo G. Madoery*†, Juan A. Fraire*‡§, Jorge M. Finochietto*†, Pedro R. D'Argenio*‡§

*Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Córdoba, Argentina
†Facultad de Ciencias Exactas, Físicas y Naturales (FCEFyN), UNC. Córdoba, Argentina
‡Facultad de Matemática, Astronomía, Física y Computación (FAMAF), UNC, Córdoba, Argentina
§Department of Computer Science, Saarland University, Saarbrücken, Germany

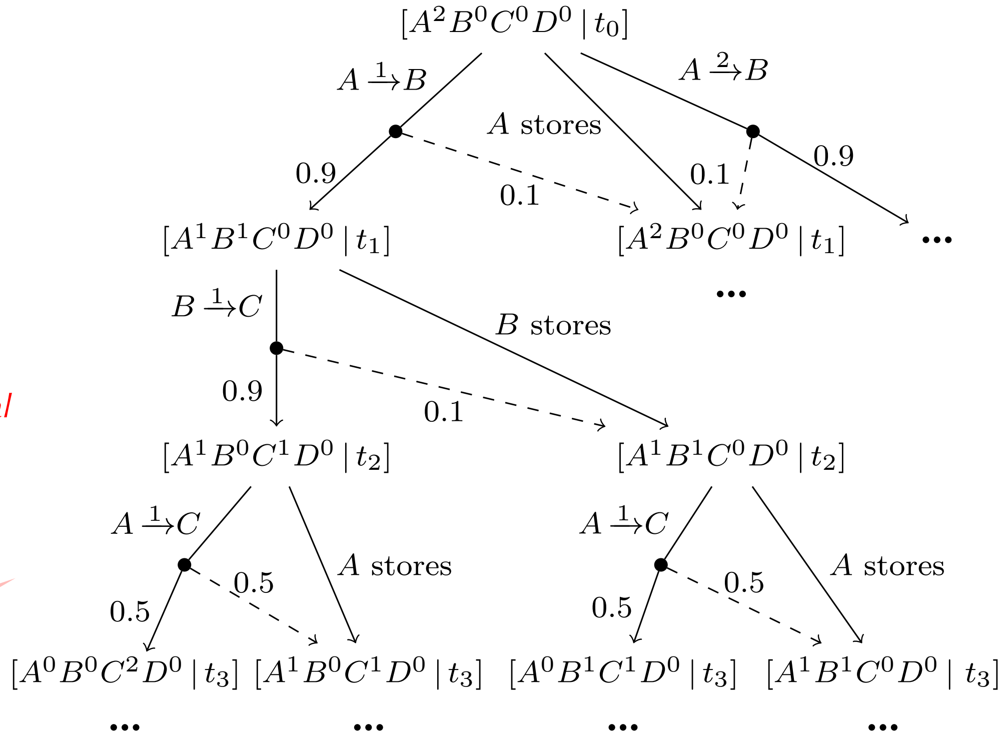
*Abstract*—Delay Tolerant Networking (DTN) has been proposed to provide efficient and autonomous store-carry-and-forward data transport for space-terrestrial networks. Since these networks relay on scheduled contact plans, Contact Graph Routing (CGR) can be used to optimize routing and data deliv-

These types of deterministic DTNs are known as scheduled DTNs and can take advantage of a *contact plan* comprising the future network connectivity in order to optimize data forwarding. However, scheduled routing solutions such as Contact Graph Routing (CGR) assumes the estimation of the future topology status is highly accurate [3]. Indeed, CGR

# Optimality through Markov Decision Processes

Bellman equations for reachability:

$$
\begin{aligned}
x_s &= 1 && \text{if } s \in \textit{Goal} \\
x_s &= 0 && \text{if } s \not\models \Diamond \textit{Goal} \\
x_s &= \max_{\alpha \in Act(s)} \sum_{t \in S} \mathbf{P}(s,\alpha,t) \cdot x_t && \text{if } s \models \Diamond \textit{Goal} \\
& && \text{and } s \notin \textit{Goal}
\end{aligned}
$$




We have a reachability problem where goal states are those with a copy at target node

UNC

# Optimality through Markov Decision Processes

$$x_s^{(0)} = 1 \qquad \text{if } s \in \textit{Goal}$$

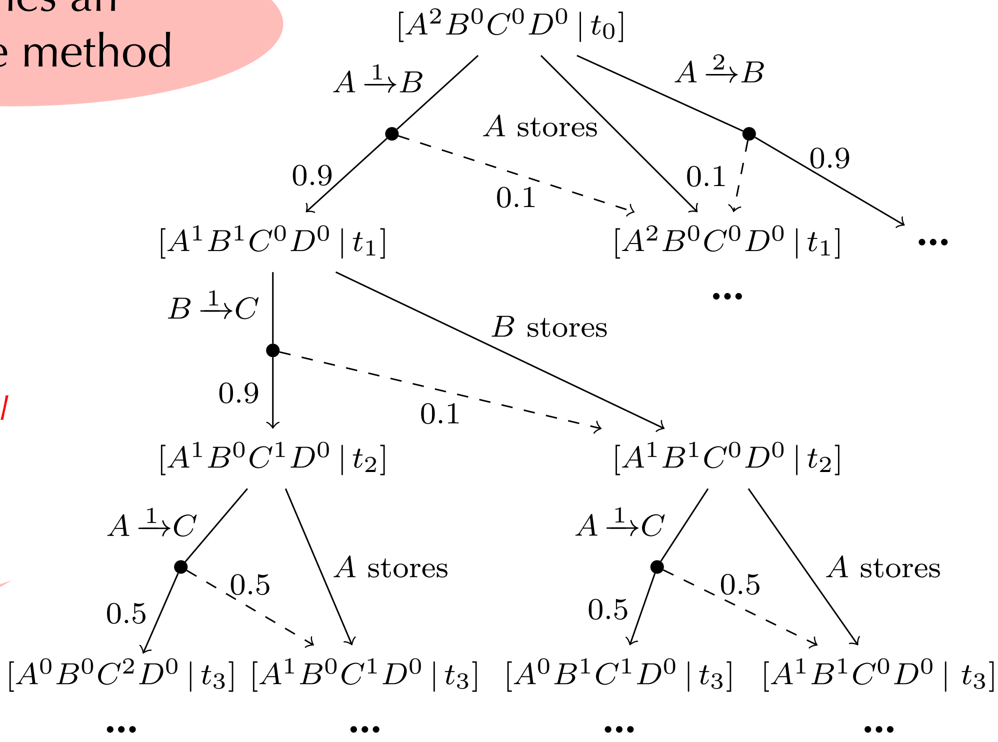
$$x_s^{(0)} = 0 \qquad \text{if } s \notin \textit{Goal}$$

Defines an iterative method

$$x_s^{(i+1)} = 1 \qquad \text{if } s \in \textit{Goal}$$

$$x_s^{(i+1)} = 0 \qquad \text{if } s \not\models \Diamond\textit{Goal}$$

$$x_s^{(i+1)} = \max_{\alpha \in Act(s)} \sum_{t \in S} \mathbf{P}(s,\alpha,t) \cdot x_t^{(i)} \qquad \text{if } s \models \Diamond\textit{Goal} \text{ and } s \notin \textit{Goal}$$




We have a reachability problem where goal states are those with a copy at target node

UNC

# Optimality through Markov Decision Processes

$$x_s^{(0)} = 1 \qquad \text{if } s \in \mathit{Goal}$$

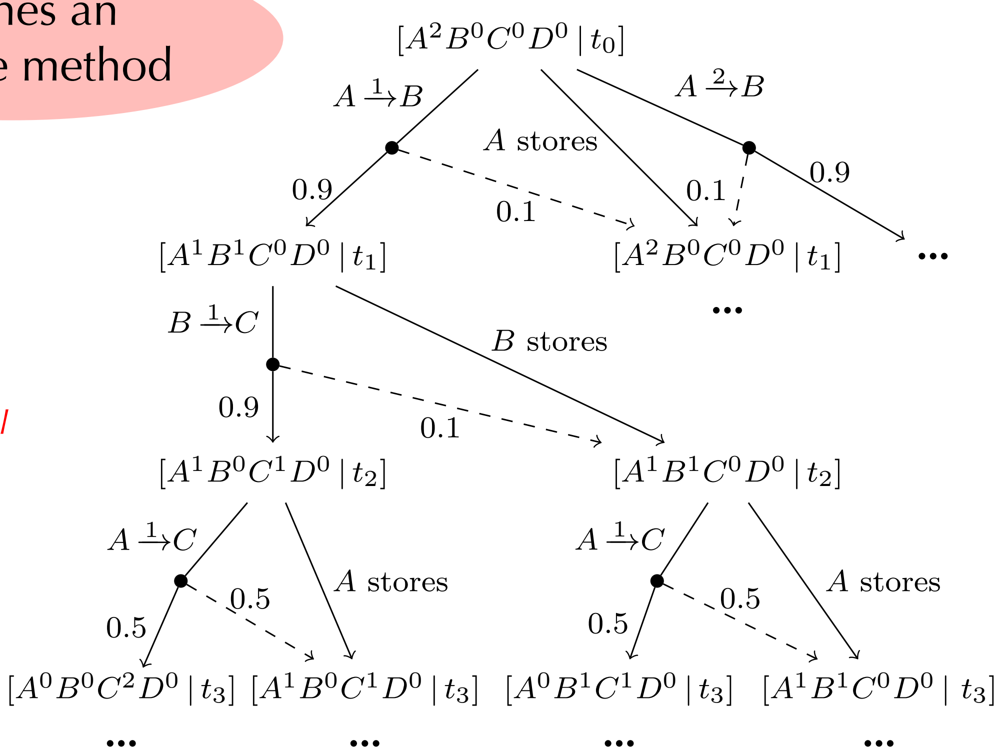
$$x_s^{(0)} = 0 \qquad \text{if } s \notin \mathit{Goal}$$

Defines an iterative method

$$x_s^{(i+1)} = 1 \qquad \text{if } s \in \mathit{Goal}$$

$$x_s^{(i+1)} = 0 \qquad \text{if } s \not\models \Diamond \mathit{Goal}$$

$$x_s^{(i+1)} = \max_{\alpha \in Act(s)} \sum_{t \in S} \mathbf{P}(s, \alpha, t) \cdot x_t^{(i)} \qquad \text{if } s \models \Diamond \mathit{Goal} \text{ and } s \notin \mathit{Goal}$$




Observe:
MDP is acyclic

CONICET

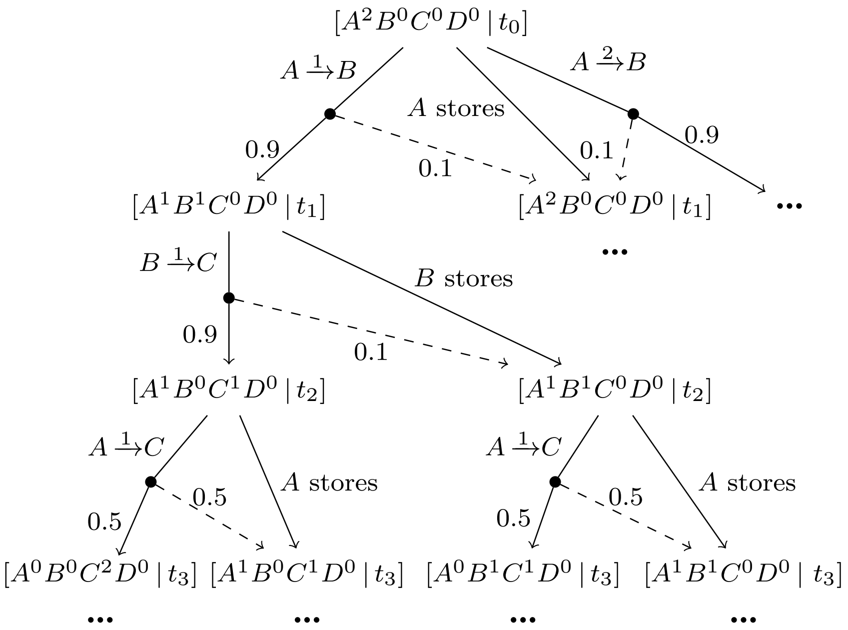
UNC

## First technique

# Routing under Uncertain Contact Plans (RUCoP)

RUCoP:

- follows Bellman equations backwardly (starting from goal states)
- only one pass required
- only maximizing subgraph (Markov chain!) is preserved
- Non-maximizing parts are discarded
- Already analyzed states are moved to disk




CONICET

Raverta et al., 2021. Ad Hoc Networks

UNC

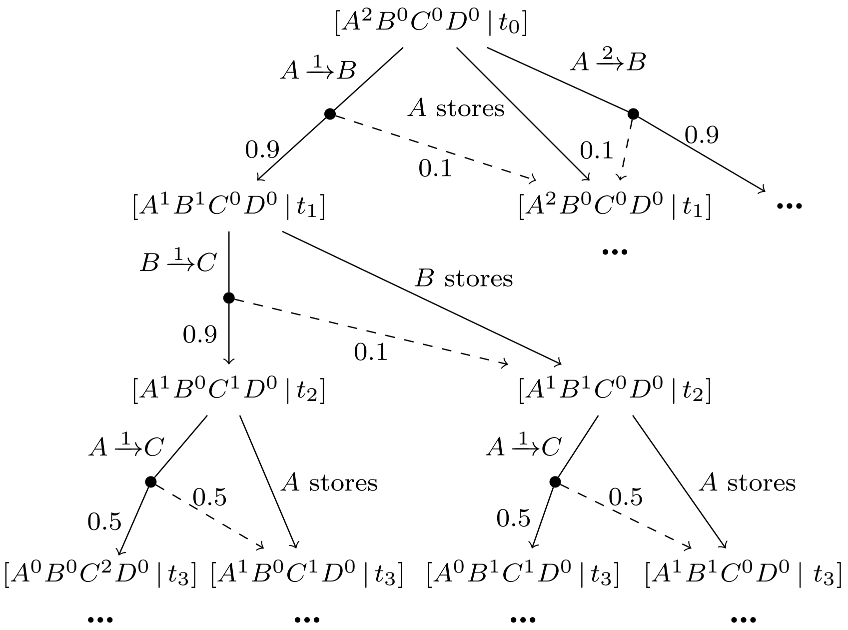
## First technique

# Routing under Uncertain Contact Plans (RUCoP)

**Algorithm 1:** The RUCoP algorithm

**Input:** Uncertain time varying graph $\mathcal{G}$, $num\_copies$, Target
**Output:** Explored states $\mathcal{S}$, Routing table $Tr$, Successful delivery probability $Pr$

determine *successful states* $\mathcal{S}_{t_{end}}$ for $num\_copies$
$\mathcal{S} \leftarrow \mathcal{S}_{t_{end}}$
**for all** $t_i \in \mathcal{T}$, starting from $t_{end-1}$ **do**
  $\mathcal{S}_{t_i} \leftarrow \varnothing$
  **for all** state $s \in \mathcal{S}_{t_{i+1}}$ **do**
    determine *carrier nodes* $\mathcal{C}_{t_i}$
    **for all** node $c \in \mathcal{C}_{t_i}$ **do**
      $\mathcal{P}_c \leftarrow \{c\} \cup \bigcup_{c' \in pred^{+}_{G_{t_i}}(c)} path_{G_{t_i}}(c', c)$
      $\mathcal{R}_c \leftarrow \left\{R \subseteq \{0, \ldots cp(c)\} \times \mathcal{P}_c \mid \sum_{(k,\rho) \in R} k = cp(c)\right\}$
    **end for**
    $Tr(s) \leftarrow \left\{\bigcup_{c \in \mathcal{C}_{t_i}} R_c \mid \forall c \in \mathcal{C}_{t_i} : R_c \in \mathcal{R}_c\right\}$
    **for all** $R \in Tr(s)$ **do**
      $s' \leftarrow get\_previous\_state(s, R)$
      $\mathcal{S}_{t_i} \leftarrow \mathcal{S}_{t_i} \cup \{s'\}$
      $pr_R \leftarrow SDP(R, s', t_i)$
      **if** $Pr(s')$ is undefined or $Pr(s') < pr_R$ **then**
        $Pr(s') \leftarrow pr_R$
        $best\_action(s') \leftarrow R$
      **end if**
    **end for**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_{t_i}$
  **end for**
**end for**
**return** $\mathcal{S}$, $Tr$, $Pr$




Raverta et al., 2021. Ad Hoc Networks

UNC

## First technique

# Routing under Uncertain Contact Plans (RUCoP)

**Algorithm 1:** The RUCoP algorithm

**Input:** Uncertain time varying graph $\mathcal{G}$, $num\_copies$, Target

**Output:** Explored states $\mathcal{S}$, Routing table $Tr$, Successful delivery probability $Pr$

determine *successful states* $\mathcal{S}_{t_{end}}$ for $num\_copies$

$\mathcal{S} \leftarrow \mathcal{S}_{t_{end}}$

**for all** $t_i \in \mathcal{T}$, starting from $t_{end-1}$ **do**

  $\mathcal{S}_{t_i} \leftarrow \varnothing$

  **for all** state $s \in \mathcal{S}_{t_{i+1}}$ **do**

    determine *carrier nodes* $\mathcal{C}_{t_i}$

    **for all** node $c \in \mathcal{C}_{t_i}$ **do**

      $\mathcal{P}_c \leftarrow \{c\} \cup \bigcup_{c' \in pred^{+}_{G_{t_i}}(c)} path_{G_{t_i}}(c', c)$

      $\mathcal{R}_c \leftarrow \left\{R \subseteq \{0, \ldots cp(c)\} \times \mathcal{P}_c \mid \sum_{(k,\rho)\in R} k = cp(c)\right\}$

    **end for**

    $Tr(s) \leftarrow \left\{\bigcup_{c\in\mathcal{C}_{t_i}} R_c \mid \forall c \in \mathcal{C}_{t_i} : R_c \in \mathcal{R}_c\right\}$

    **for all** $R \in Tr(s)$ **do**

      $s' \leftarrow get\_previous\_state(s, R)$

      $\mathcal{S}_{t_i} \leftarrow \mathcal{S}_{t_i} \cup \{s'\}$

      $pr_R \leftarrow SDP(R, s', t_i)$

      **if** $Pr(s')$ is undefined or $Pr(s') < pr_R$ **then**

        $Pr(s') \leftarrow pr_R$

        $best\_action(s') \leftarrow R$

      **end if**

    **end for**

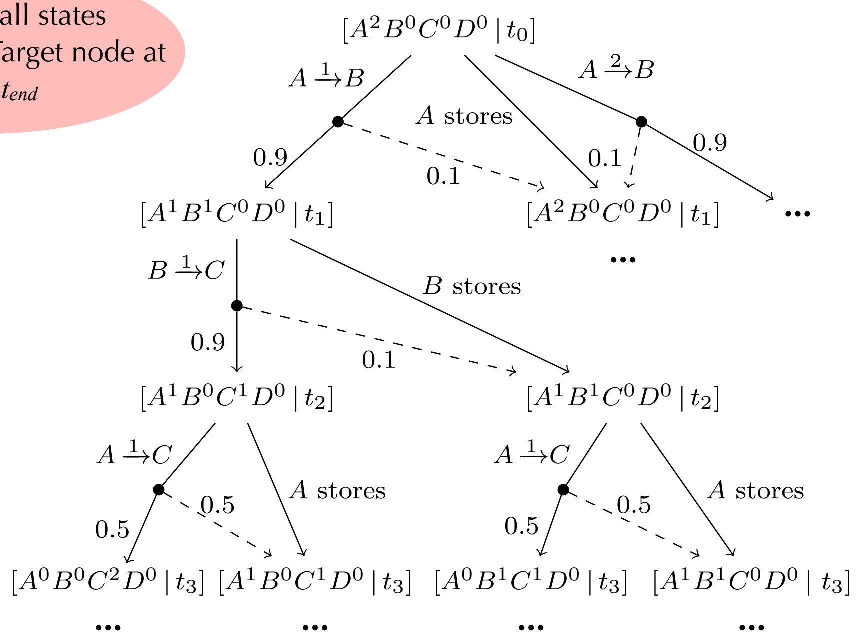
    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_{t_i}$

  **end for**

**end for**

**return** $\mathcal{S}$, $Tr$, $Pr$

Start from all states that reach the Target node at time $t_{end}$




$$\left\{ [A^w B^x C^y D^z \mid t_4] \;\middle|\; z > 1 \wedge w + x + y + z = 2 \right\}$$

Raverta et al., 2021. Ad Hoc Networks

First technique

# Routing under Uncertain Contact Plans (RUCoP)

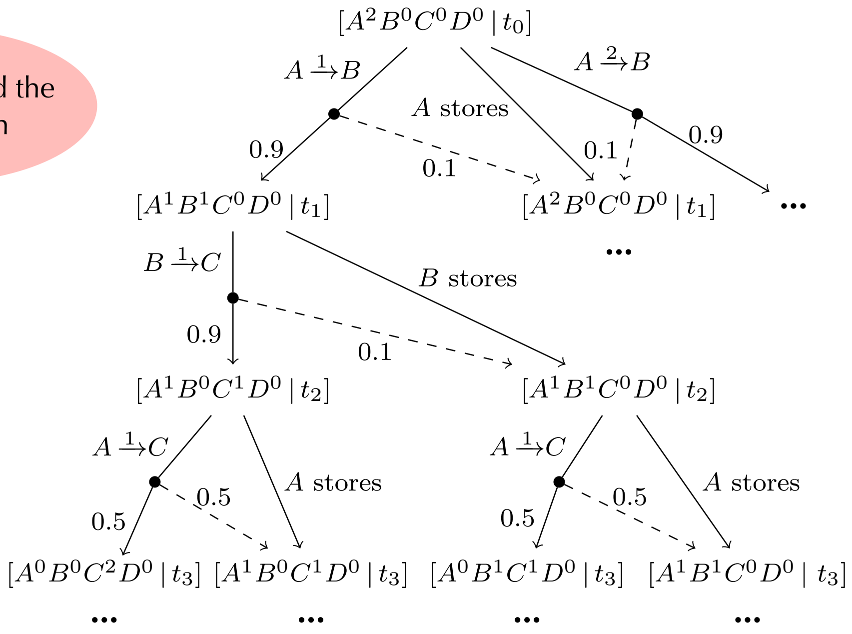
**Algorithm 1:** The RUCoP algorithm

**Input:** Uncertain time varying graph $\mathcal{G}$, *num_copies*, Target

**Output:** Explored states $\mathcal{S}$, Routing table $Tr$, Successful delivery probability $Pr$

determine *successful states* $\mathcal{S}_{t_{end}}$ for *num_copies*
$\mathcal{S} \leftarrow \mathcal{S}_{t_{end}}$
**for all** $t_i \in \mathcal{T}$, starting from $t_{end-1}$ **do**
  $\mathcal{S}_{t_i} \leftarrow \varnothing$
  **for all** state $s \in \mathcal{S}_{t_{i+1}}$ **do**
    determine *carrier nodes* $\mathcal{C}_{t_i}$
    **for all** node $c \in \mathcal{C}_{t_i}$ **do**
      $\mathcal{P}_c \leftarrow \{c\} \cup \bigcup_{c' \in pred^+_{G_{t_i}}(c)} path_{G_{t_i}}(c', c)$
      $\mathcal{R}_c \leftarrow \left\{R \subseteq \{0, \ldots cp(c)\} \times \mathcal{P}_c \mid \sum_{(k,\rho) \in R} k = cp(c)\right\}$
    **end for**
    $Tr(s) \leftarrow \left\{\bigcup_{c \in \mathcal{C}_{t_i}} R_c \mid \forall c \in \mathcal{C}_{t_i} : R_c \in \mathcal{R}_c\right\}$
    **for all** $R \in Tr(s)$ **do**
      $s' \leftarrow get\_previous\_state(s, R)$
      $\mathcal{S}_{t_i} \leftarrow \mathcal{S}_{t_i} \cup \{s'\}$
      $pr_R \leftarrow SDP(R, s', t_i)$
      **if** $Pr(s')$ is undefined or $Pr(s') < pr_R$ **then**
        $Pr(s') \leftarrow pr_R$
        $best\_action(s') \leftarrow R$
      **end if**
    **end for**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_{t_i}$
  **end for**
**end for**
**return** $\mathcal{S}$, $Tr$, $Pr$

Travel backward the Contact Plan




$$\left\{ [A^w B^x C^y D^z \mid t_4] \;\middle|\; z > 1 \wedge w + x + y + z = 2 \right\}$$

Raverta et al., 2021. Ad Hoc Networks

UNC

## First technique

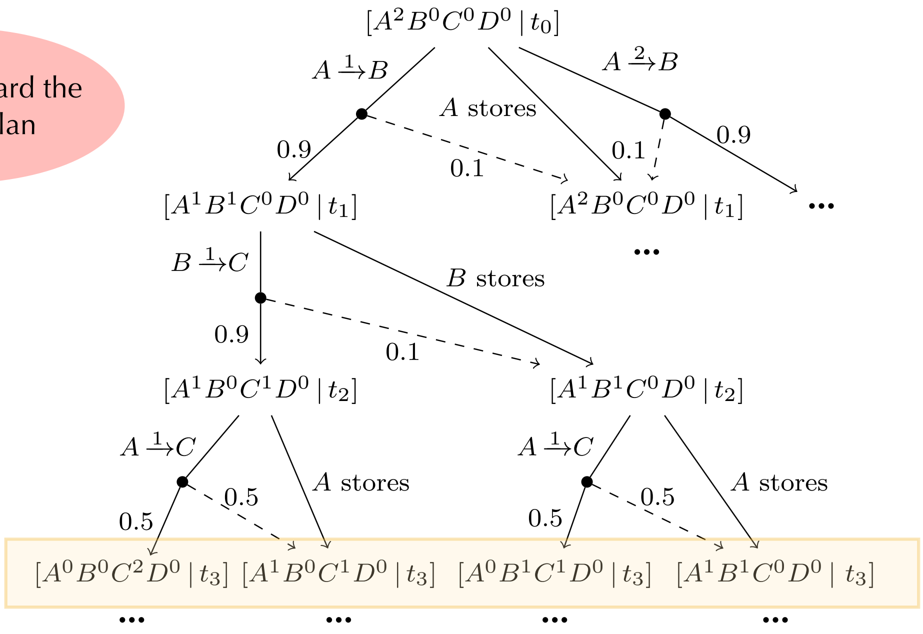
# Routing under Uncertain Contact Plans (RUCoP)

**Algorithm 1:** The RUCoP algorithm

**Input:** Uncertain time varying graph $\mathcal{G}$, $num\_copies$, Target
**Output:** Explored states $\mathcal{S}$, Routing table $Tr$, Successful delivery probability $Pr$

determine *successful states* $\mathcal{S}_{t_{end}}$ for $num\_copies$
$\mathcal{S} \leftarrow \mathcal{S}_{t_{end}}$
**for all** $t_i \in \mathcal{T}$, starting from $t_{end-1}$ **do**
  $\mathcal{S}_{t_i} \leftarrow \varnothing$
  **for all** state $s \in \mathcal{S}_{t_{i+1}}$ **do**
    determine *carrier nodes* $\mathcal{C}_{t_i}$
    **for all** node $c \in \mathcal{C}_{t_i}$ **do**
      $\mathcal{P}_c \leftarrow \{c\} \cup \bigcup_{c' \in pred^{+}_{G_{t_i}}(c)} path_{G_{t_i}}(c', c)$
      $\mathcal{R}_c \leftarrow \{R \subseteq \{0, \dots cp(c)\} \times \mathcal{P}_c \mid \sum_{(k,\rho)\in R} k = cp(c)\}$
    **end for**
    $Tr(s) \leftarrow \{\bigcup_{c \in \mathcal{C}_{t_i}} R_c \mid \forall c \in \mathcal{C}_{t_i} : R_c \in \mathcal{R}_c\}$
    **for all** $R \in Tr(s)$ **do**
      $s' \leftarrow get\_previous\_state(s, R)$
      $\mathcal{S}_{t_i} \leftarrow \mathcal{S}_{t_i} \cup \{s'\}$
      $pr_R \leftarrow SDP(R, s', t_i)$
      **if** $Pr(s')$ is undefined or $Pr(s') < pr_R$ **then**
        $Pr(s') \leftarrow pr_R$
        $best\_action(s') \leftarrow R$
      **end if**
    **end for**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_{t_i}$
  **end for**
**end for**
**return** $\mathcal{S}$, $Tr$, $Pr$

Travel backward the Contact Plan




$$\left\{ [A^w B^x C^y D^z \mid t_4] \;\middle|\; z > 1 \wedge w + x + y + z = 2 \right\}$$

Raverta et al., 2021. Ad Hoc Networks

First technique

# Routing under Uncertain Contact Plans (RUCoP)

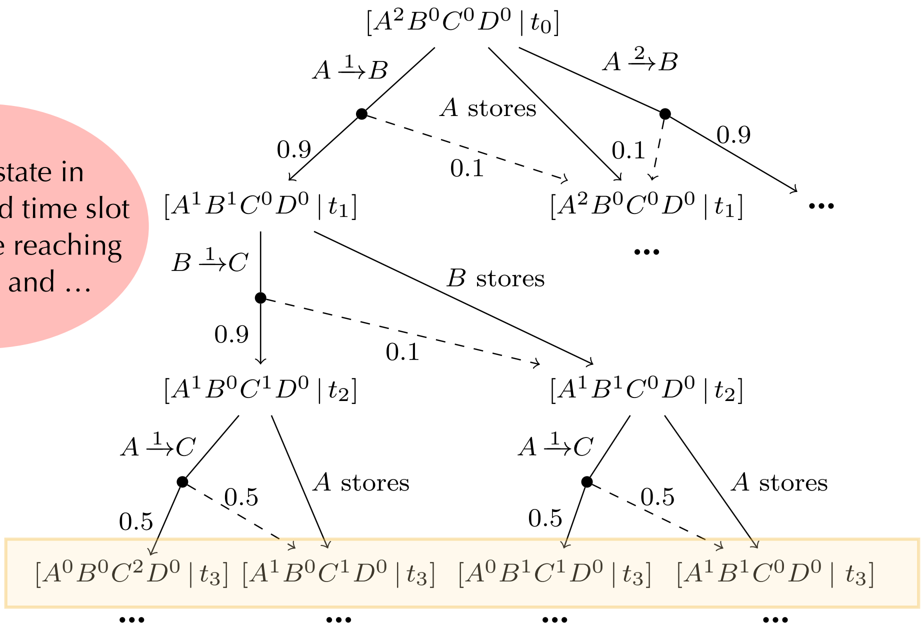
**Algorithm 1:** The RUCoP algorithm

**Input:** Uncertain time varying graph $\mathcal{G}$, *num_copies*, Target

**Output:** Explored states $\mathcal{S}$, Routing table $Tr$, Successful delivery probability $Pr$

determine *successful states* $\mathcal{S}_{t_{end}}$ for *num_copies*
$\mathcal{S} \leftarrow \mathcal{S}_{t_{end}}$
**for all** $t_i \in \mathcal{T}$, starting from $t_{end-1}$ **do**
  $\mathcal{S}_{t_i} \leftarrow \varnothing$
  **for all** state $s \in \mathcal{S}_{t_{i+1}}$ **do**
    determine *carrier nodes* $\mathcal{C}_{t_i}$
    **for all** node $c \in \mathcal{C}_{t_i}$ **do**
      $\mathcal{P}_c \leftarrow \{c\} \cup \bigcup_{c' \in pred^+_{G_{t_i}}(c)} path_{G_{t_i}}(c', c)$
      $\mathcal{R}_c \leftarrow \left\{ R \subseteq \{0, \ldots cp(c)\} \times \mathcal{P}_c \mid \sum_{(k,\rho)\in R} k = cp(c) \right\}$
    **end for**
    $Tr(s) \leftarrow \left\{ \bigcup_{c \in \mathcal{C}_{t_i}} R_c \mid \forall c \in \mathcal{C}_{t_i} : R_c \in \mathcal{R}_c \right\}$
    **for all** $R \in Tr(s)$ **do**
      $s' \leftarrow get\_previous\_state(s, R)$
      $\mathcal{S}_{t_i} \leftarrow \mathcal{S}_{t_i} \cup \{s'\}$
      $pr_R \leftarrow SDP(R, s', t_i)$
      **if** $Pr(s')$ is undefined or $Pr(s') < pr_R$ **then**
        $Pr(s') \leftarrow pr_R$
        $best\_action(s') \leftarrow R$
      **end if**
    **end for**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_{t_i}$
  **end for**
**end for**
**return** $\mathcal{S}$, $Tr$, $Pr$

For each state in the last visited time slot construct the reaching transitions and …




$$\left\{ [A^wB^xC^yD^z \mid t_4] \;\middle|\; z > 1 \land w + x + y + z = 2 \right\}$$

Raverta et al., 2021. Ad Hoc Networks

UNC

First technique

# Routing under Uncertain Contact Plans (RUCoP)

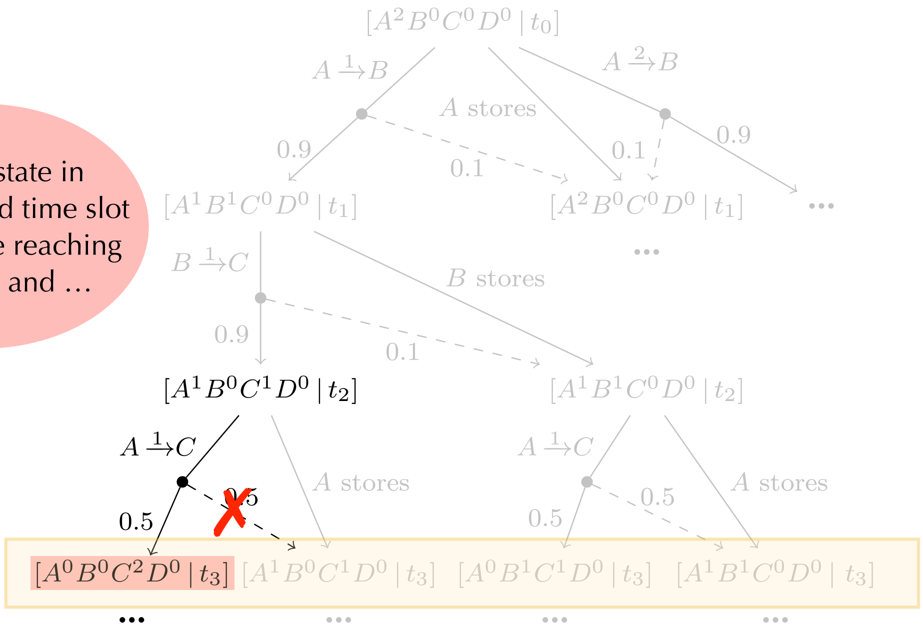
**Algorithm 1:** The RUCoP algorithm

**Input:** Uncertain time varying graph $\mathcal{G}$, $num\_copies$, Target

**Output:** Explored states $\mathcal{S}$, Routing table $Tr$, Successful delivery probability $Pr$

```
determine successful states S_{t_end} for num_copies
S ← S_{t_end}
for all t_i ∈ T, starting from t_{end-1} do
   S_{t_i} ← ∅
   for all state s ∈ S_{t_{i+1}} do
      determine carrier nodes C_{t_i}
      for all node c ∈ C_{t_i} do
         P_c ← {c} ∪ ⋃_{c' ∈ pred^+_{G_{t_i}}(c)} path_{G_{t_i}}(c', c)
         R_c ← {R ⊆ {0, ... cp(c)} × P_c | Σ_{(k,ρ)∈R} k = cp(c)}
      end for
      Tr(s) ← {⋃_{c∈C_{t_i}} R_c | ∀c ∈ C_{t_i} : R_c ∈ R_c}
      for all R ∈ Tr(s) do
         s' ← get_previous_state(s, R)
         S_{t_i} ← S_{t_i} ∪ {s'}
         pr_R ← SDP(R, s', t_i)
         if Pr(s') is undefined or Pr(s') < pr_R then
            Pr(s') ← pr_R
            best_action(s') ← R
         end if
      end for
      S ← S ∪ S_{t_i}
   end for
end for
return S, Tr, Pr
```

For each state in the last visited time slot construct the reaching transitions and …




$$\left\{ [A^wB^xC^yD^z \mid t_4] \;\middle|\; z > 1 \wedge w + x + y + z = 2 \right\}$$

First technique

# Routing under Uncertain Contact Plans (RUCoP)

**Algorithm 1:** The RUCoP algorithm

**Input:** Uncertain time varying graph $\mathcal{G}$, $num\_copies$, Target

**Output:** Explored states $\mathcal{S}$, Routing table $Tr$, Successful delivery probability $Pr$

determine *successful states* $\mathcal{S}_{t_{end}}$ for $num\_copies$

$\mathcal{S} \leftarrow \mathcal{S}_{t_{end}}$

**for all** $t_i \in \mathcal{T}$, starting from $t_{end-1}$ **do**

  $\mathcal{S}_{t_i} \leftarrow \varnothing$

  **for all** state $s \in \mathcal{S}_{t_{i+1}}$ **do**

    determine *carrier nodes* $\mathcal{C}_{t_i}$

    **for all** node $c \in \mathcal{C}_{t_i}$ **do**

      $\mathcal{P}_c \leftarrow \{c\} \cup \bigcup_{c' \in pred^+_{G_{t_i}}(c)} path_{G_{t_i}}(c', c)$

      $\mathcal{R}_c \leftarrow \left\{R \subseteq \{0, \ldots cp(c)\} \times \mathcal{P}_c \mid \sum_{(k,\rho)\in R} k = cp(c)\right\}$

    **end for**

    $Tr(s) \leftarrow \left\{\bigcup_{c \in \mathcal{C}_{t_i}} R_c \mid \forall c \in \mathcal{C}_{t_i} : R_c \in \mathcal{R}_c\right\}$

    **for all** $R \in Tr(s)$ **do**

      $s' \leftarrow get\_previous\_state(s, R)$

      $\mathcal{S}_{t_i} \leftarrow \mathcal{S}_{t_i} \cup \{s'\}$

      $pr_R \leftarrow SDP(R, s', t_i)$

      **if** $Pr(s')$ is undefined or $Pr(s') < pr_R$ **then**

        $Pr(s') \leftarrow pr_R$

        $best\_action(s') \leftarrow R$

      **end if**

    **end for**

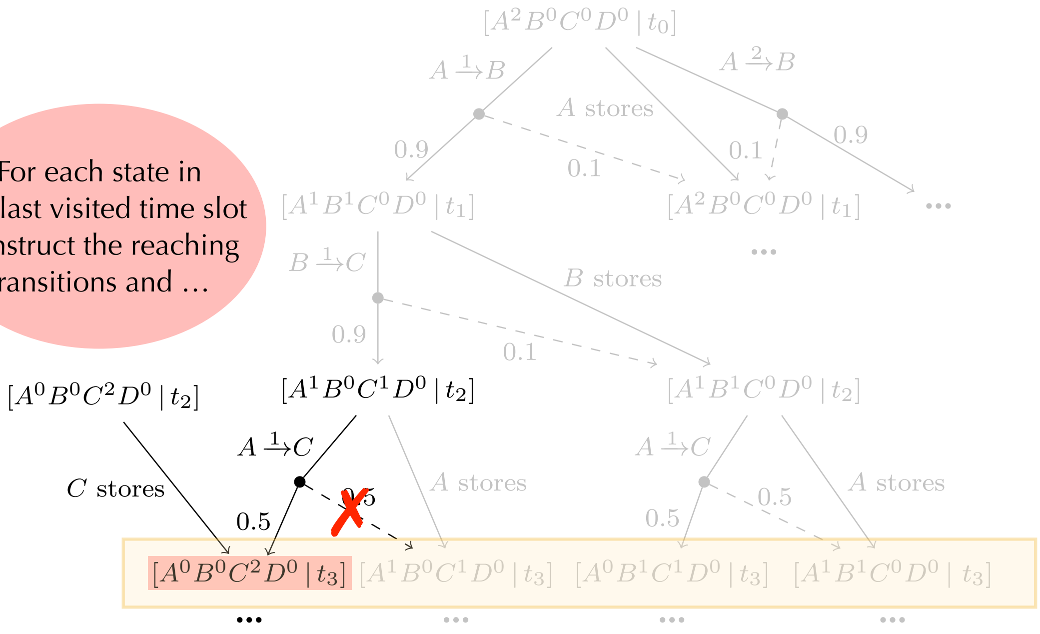
    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_{t_i}$

  **end for**

**end for**

**return** $\mathcal{S}$, $Tr$, $Pr$

For each state in the last visited time slot construct the reaching transitions and …




$$\left\{ [A^w B^x C^y D^z \,|\, t_4] \;\middle|\; z > 1 \wedge w + x + y + z = 2 \right\}$$

Raverta et al., 2021. Ad Hoc Networks

UNC

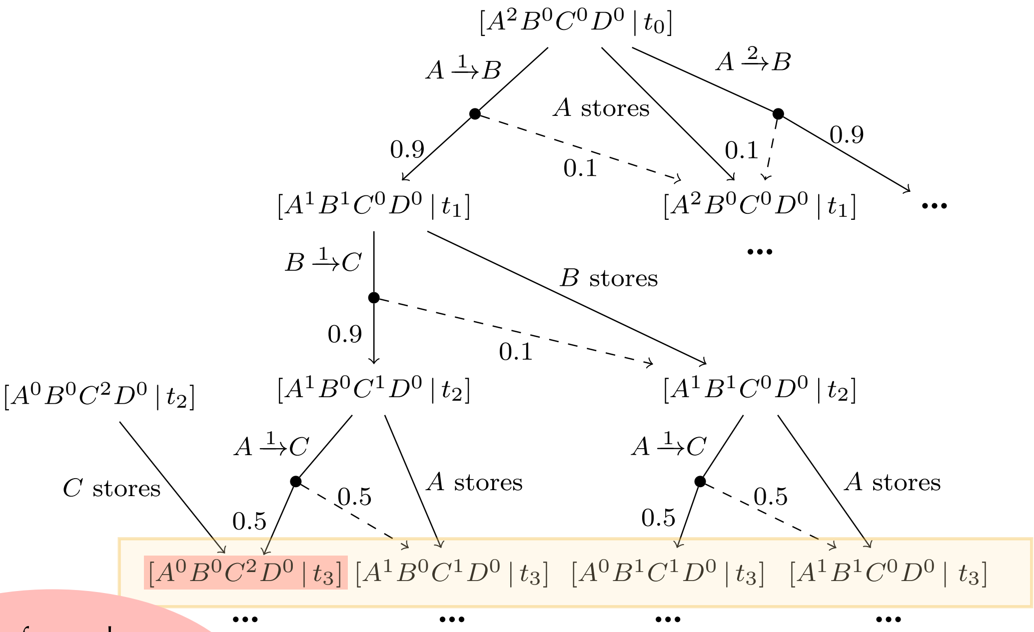
## First technique

# Routing under Uncertain Contact Plans (RUCoP)

**Algorithm 1:** The RUCoP algorithm

**Input:** Uncertain time varying graph $\mathcal{G}$, *num_copies*, Target
**Output:** Explored states $\mathcal{S}$, Routing table $Tr$, Successful delivery probability $Pr$

```
determine successful states S_{t_end} for num_copies
S ← S_{t_end}
for all t_i ∈ T, starting from t_{end−1} do
  S_{t_i} ← ∅
  for all state s ∈ S_{t_{i+1}} do
    determine carrier nodes C_{t_i}
    for all node c ∈ C_{t_i} do
      P_c ← {c} ∪ ⋃_{c' ∈ pred^+_{G_{t_i}}(c)} path_{G_{t_i}}(c', c)
      R_c ← {R ⊆ {0, … cp(c)} × P_c | Σ_{(k,ρ)∈R} k = cp(c)}
    end for
    Tr(s) ← {⋃_{c∈C_{t_i}} R_c | ∀c ∈ C_{t_i} : R_c ∈ R_c}
    for all R ∈ Tr(s) do
      s' ← get_previous_state(s, R)
      S_{t_i} ← S_{t_i} ∪ {s'}
      pr_R ← SDP(R, s', t_i)
      if Pr(s') is undefined or Pr(s') < pr_R then
        Pr(s') ← pr_R
        best_action(s') ← R
      end if
    end for
    S ← S ∪ S_{t_i}
  end for
end for
return S, Tr, Pr
```




… for each transition update the source state probability value and its respective optimizing transition

$$\left\{ [A^w B^x C^y D^z \mid t_4] \;\middle|\; z > 1 \wedge w + x + y + z = 2 \right\}$$

Raverta et al., 2021. Ad Hoc Networks

First technique

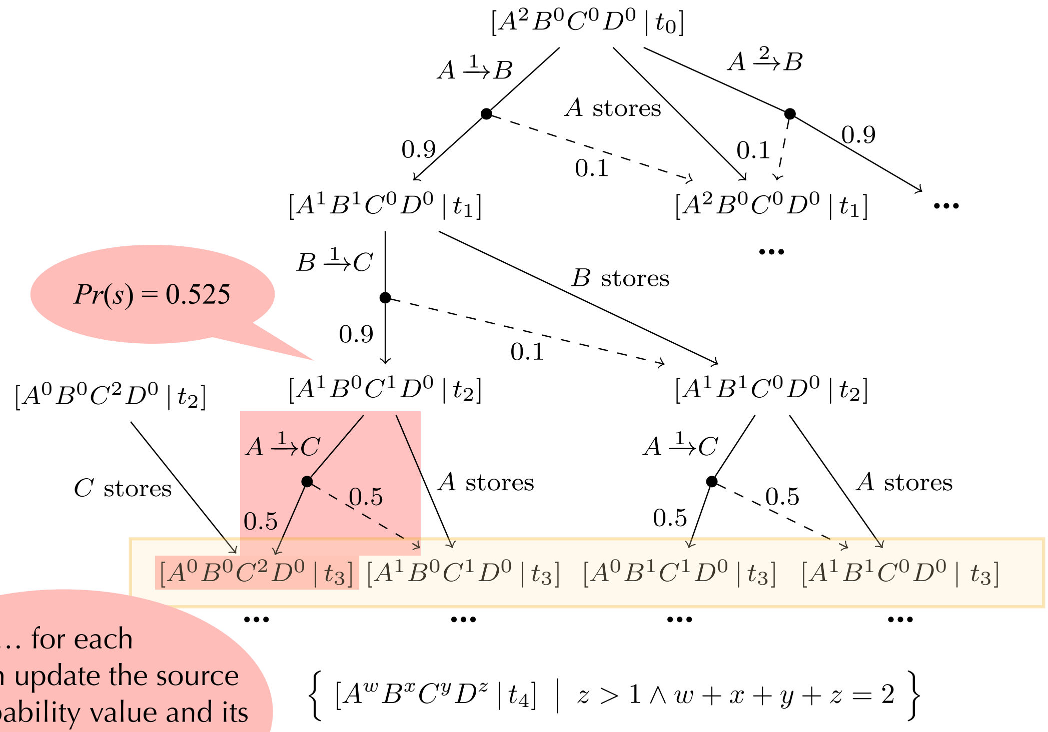
# Routing under Uncertain Contact Plans (RUCoP)

**Algorithm 1:** The RUCoP algorithm

**Input:** Uncertain time varying graph $\mathcal{G}$, $num\_copies$, Target

**Output:** Explored states $\mathcal{S}$, Routing table $Tr$, Successful delivery probability $Pr$

determine *successful states* $\mathcal{S}_{t_{end}}$ for $num\_copies$
$\mathcal{S} \leftarrow \mathcal{S}_{t_{end}}$
**for all** $t_i \in \mathcal{T}$, starting from $t_{end-1}$ **do**
  $\mathcal{S}_{t_i} \leftarrow \varnothing$
  **for all** state $s \in \mathcal{S}_{t_{i+1}}$ **do**
    determine *carrier nodes* $\mathcal{C}_{t_i}$
    **for all** node $c \in \mathcal{C}_{t_i}$ **do**
      $\mathcal{P}_c \leftarrow \{c\} \cup \bigcup_{c' \in pred^+_{G_{t_i}}(c)} path_{G_{t_i}}(c', c)$
      $\mathcal{R}_c \leftarrow \left\{ R \subseteq \{0, \ldots cp(c)\} \times \mathcal{P}_c \mid \sum_{(k,\rho) \in R} k = cp(c) \right\}$
    **end for**
    $Tr(s) \leftarrow \left\{ \bigcup_{c \in \mathcal{C}_{t_i}} R_c \mid \forall c \in \mathcal{C}_{t_i} : R_c \in \mathcal{R}_c \right\}$
    **for all** $R \in Tr(s)$ **do**
      $s' \leftarrow get\_previous\_state(s, R)$
      $\mathcal{S}_{t_i} \leftarrow \mathcal{S}_{t_i} \cup \{s'\}$
      $pr_R \leftarrow SDP(R, s', t_i)$
      **if** $Pr(s')$ is undefined or $Pr(s') < pr_R$ **then**
        $Pr(s') \leftarrow pr_R$
        $best\_action(s') \leftarrow R$
      **end if**
    **end for**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_{t_i}$
  **end for**
**end for**
**return** $\mathcal{S}$, $Tr$, $Pr$




$Pr(s) = 0.525$

… for each transition update the source state probability value and its respective optimizing transition

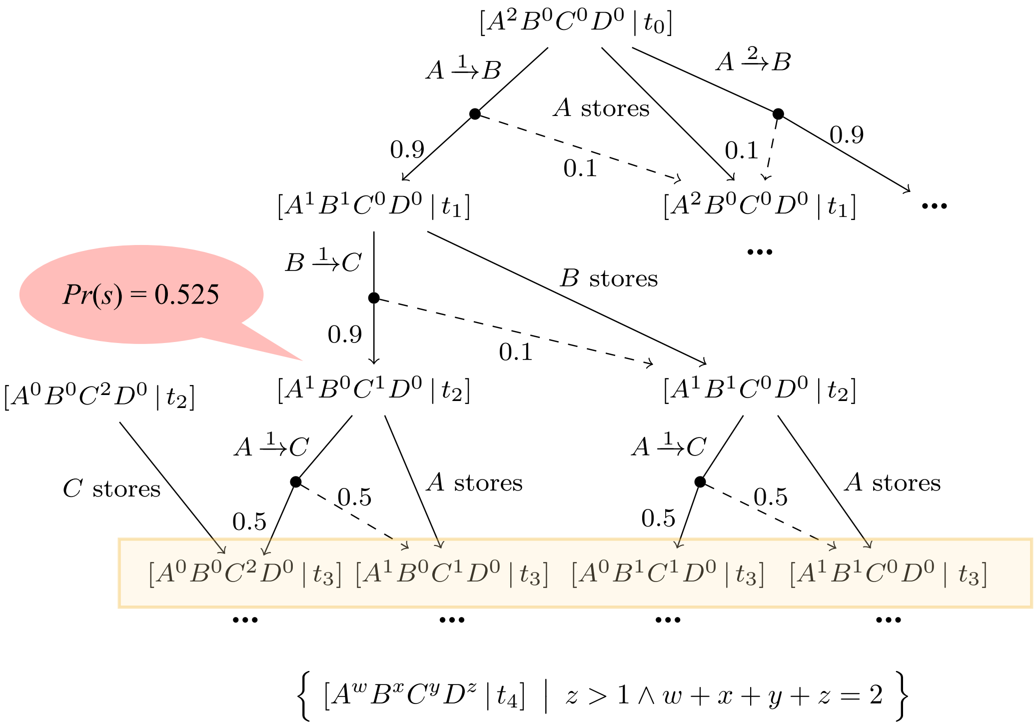
First technique

# Routing under Uncertain Contact Plans (RUCoP)

**Algorithm 1:** The RUCoP algorithm

**Input:** Uncertain time varying graph $\mathcal{G}$, *num_copies*, Target
**Output:** Explored states $\mathcal{S}$, Routing table $Tr$, Successful delivery probability $Pr$

determine *successful states* $\mathcal{S}_{t_{end}}$ for *num_copies*
$\mathcal{S} \leftarrow \mathcal{S}_{t_{end}}$
**for all** $t_i \in \mathcal{T}$, starting from $t_{end-1}$ **do**
  $\mathcal{S}_{t_i} \leftarrow \emptyset$
  **for all** state $s \in \mathcal{S}_{t_{i+1}}$ **do**
    determine *carrier nodes* $\mathcal{C}_{t_i}$
    **for all** node $c \in \mathcal{C}_{t_i}$ **do**
      $\mathcal{P}_c \leftarrow \{c\} \cup \bigcup_{c' \in pred^+_{G_{t_i}}(c)} path_{G_{t_i}}(c', c)$
      $\mathcal{R}_c \leftarrow \left\{ R \subseteq \{0, \ldots cp(c)\} \times \mathcal{P}_c \mid \sum_{(k,\rho) \in R} k = cp(c) \right\}$
    **end for**
    $Tr(s) \leftarrow \left\{ \bigcup_{c \in \mathcal{C}_{t_i}} R_c \mid \forall c \in \mathcal{C}_{t_i} : R_c \in \mathcal{R}_c \right\}$
    **for all** $R \in Tr(s)$ **do**
      $s' \leftarrow get\_previous\_state(s, R)$
      $\mathcal{S}_{t_i} \leftarrow \mathcal{S}_{t_i} \cup \{s'\}$
      $pr_R \leftarrow SDP(R, s', t_i)$
      **if** $Pr(s')$ is undefined or $Pr(s') < pr_R$ **then**
        $Pr(s') \leftarrow pr_R$
        $best\_action(s') \leftarrow R$
      **end if**
    **end for**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_{t_i}$
  **end for**
**end for**
**return** $\mathcal{S}$, $Tr$, $Pr$



First technique

# Routing under Uncertain Contact Plans (RUCoP)

**Algorithm 1:** The RUCoP algorithm

**Input:** Uncertain time varying graph $\mathcal{G}$, $num\_copies$, Target

**Output:** Explored states $\mathcal{S}$, Routing table $Tr$, Successful delivery probability $Pr$

determine *successful states* $\mathcal{S}_{t_{end}}$ for $num\_copies$

$\mathcal{S} \leftarrow \mathcal{S}_{t_{end}}$

**for all** $t_i \in \mathcal{T}$, starting from $t_{end-1}$ **do**

  $\mathcal{S}_{t_i} \leftarrow \varnothing$

  **for all** state $s \in \mathcal{S}_{t_{i+1}}$ **do**

    determine *carrier nodes* $\mathcal{C}_{t_i}$

    **for all** node $c \in \mathcal{C}_{t_i}$ **do**

      $\mathcal{P}_c \leftarrow \{c\} \cup \bigcup_{c' \in pred^+_{G_{t_i}}(c)} path_{G_{t_i}}(c', c)$

      $\mathcal{R}_c \leftarrow \left\{R \subseteq \{0, \dots cp(c)\} \times \mathcal{P}_c \mid \sum_{(k,\rho) \in R} k = cp(c)\right\}$

    **end for**

    $Tr(s) \leftarrow \left\{\bigcup_{c \in \mathcal{C}_{t_i}} R_c \mid \forall c \in \mathcal{C}_{t_i} : R_c \in \mathcal{R}_c\right\}$

    **for all** $R \in Tr(s)$ **do**

      $s' \leftarrow get\_previous\_state(s, R)$

      $\mathcal{S}_{t_i} \leftarrow \mathcal{S}_{t_i} \cup \{s'\}$

      $pr_R \leftarrow SDP(R, s', t_i)$

      **if** $Pr(s')$ is undefined or $Pr(s') < pr_R$ **then**

        $Pr(s') \leftarrow pr_R$

        $best\_action(s') \leftarrow R$

      **end if**

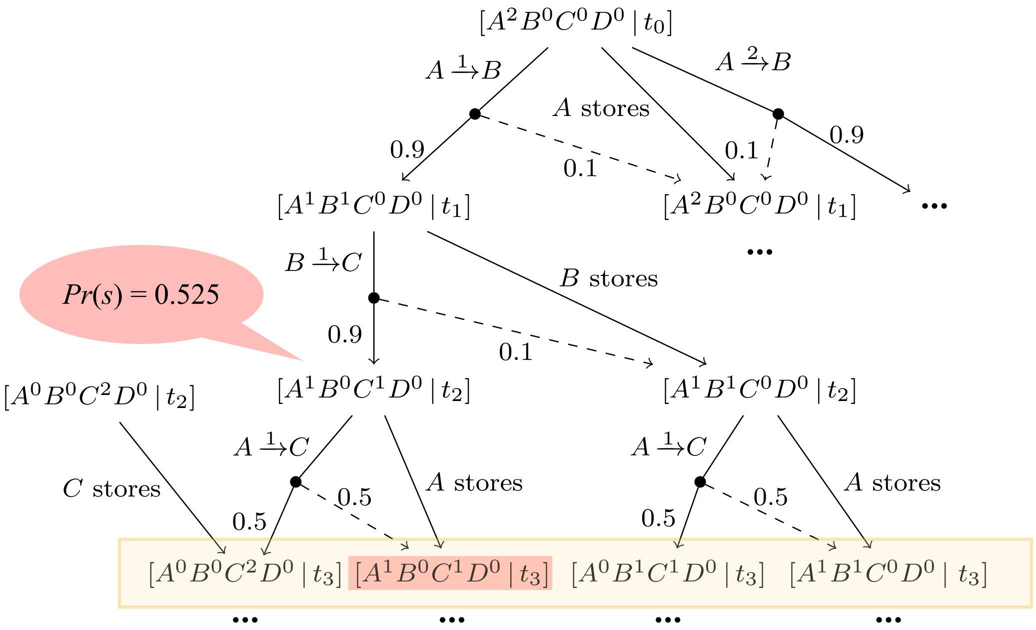
    **end for**

    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_{t_i}$

  **end for**

**end for**

**return** $\mathcal{S}$, $Tr$, $Pr$




$$\left\{ [A^w B^x C^y D^z \mid t_4] \;\middle|\; z > 1 \land w + x + y + z = 2 \right\}$$

Raverta et al., 2021. Ad Hoc Networks

UNC

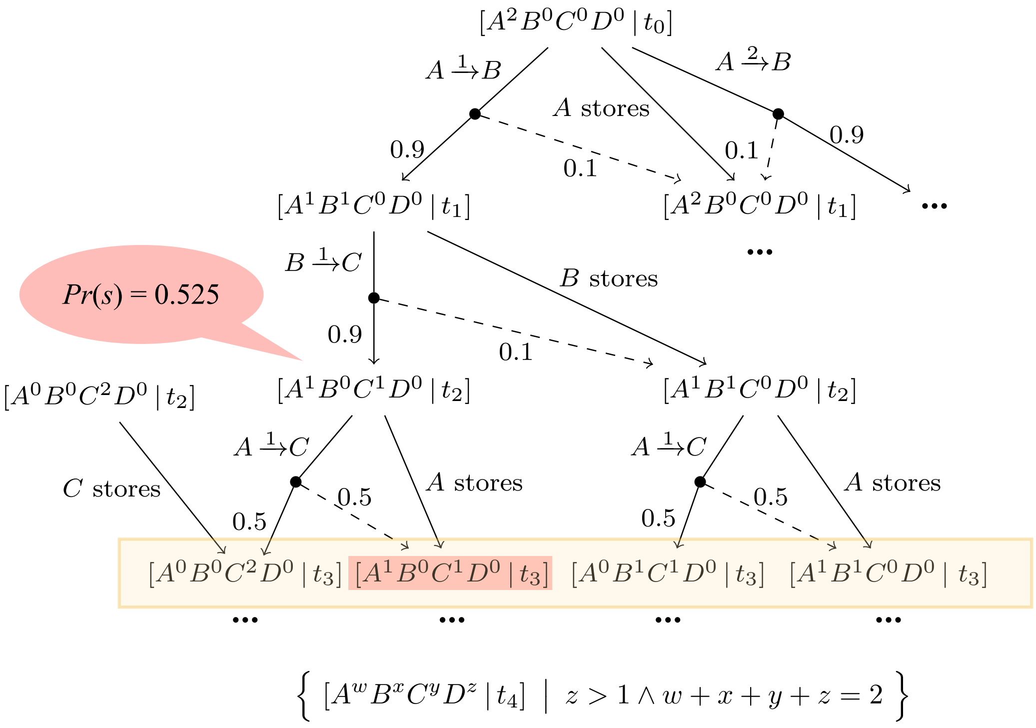
First technique

# Routing under Uncertain Contact Plans (RUCoP)

**Algorithm 1:** The RUCoP algorithm

**Input:** Uncertain time varying graph $\mathcal{G}$, $num\_copies$, Target
**Output:** Explored states $\mathcal{S}$, Routing table $Tr$, Successful delivery probability $Pr$

determine *successful states* $\mathcal{S}_{t_{end}}$ for $num\_copies$
$\mathcal{S} \leftarrow \mathcal{S}_{t_{end}}$
**for all** $t_i \in \mathcal{T}$, starting from $t_{end-1}$ **do**
  $\mathcal{S}_{t_i} \leftarrow \varnothing$
  **for all** state $s \in \mathcal{S}_{t_{i+1}}$ **do**
    determine *carrier nodes* $\mathcal{C}_{t_i}$
    **for all** node $c \in \mathcal{C}_{t_i}$ **do**
      $\mathcal{P}_c \leftarrow \{c\} \cup \bigcup_{c' \in pred^{+}_{G_{t_i}}(c)} path_{G_{t_i}}(c', c)$
      $\mathcal{R}_c \leftarrow \left\{R \subseteq \{0, \ldots cp(c)\} \times \mathcal{P}_c \mid \sum_{(k,\rho)\in R} k = cp(c)\right\}$
    **end for**
    $Tr(s) \leftarrow \left\{ \bigcup_{c \in \mathcal{C}_{t_i}} R_c \mid \forall c \in \mathcal{C}_{t_i} : R_c \in \mathcal{R}_c \right\}$
    **for all** $R \in Tr(s)$ **do**
      $s' \leftarrow get\_previous\_state(s, R)$
      $\mathcal{S}_{t_i} \leftarrow \mathcal{S}_{t_i} \cup \{s'\}$
      $pr_R \leftarrow SDP(R, s', t_i)$
      **if** $Pr(s')$ is undefined or $Pr(s') < pr_R$ **then**
        $Pr(s') \leftarrow pr_R$
        $best\_action(s') \leftarrow R$
      **end if**
    **end for**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_{t_i}$
  **end for**
**end for**
**return** $\mathcal{S}$, $Tr$, $Pr$




Raverta et al., 2021. Ad Hoc Networks

UNC

First technique

# Routing under Uncertain Contact Plans (RUCoP)

**Algorithm 1:** The RUCoP algorithm

**Input:** Uncertain time varying graph $\mathcal{G}$, $num\_copies$, Target

**Output:** Explored states $\mathcal{S}$, Routing table $Tr$, Successful delivery probability $Pr$

determine *successful states* $\mathcal{S}_{t_{end}}$ for $num\_copies$

$\mathcal{S} \leftarrow \mathcal{S}_{t_{end}}$

**for all** $t_i \in \mathcal{T}$, starting from $t_{end-1}$ **do**

  $\mathcal{S}_{t_i} \leftarrow \varnothing$

  **for all** state $s \in \mathcal{S}_{t_{i+1}}$ **do**

    determine *carrier nodes* $\mathcal{C}_{t_i}$

    **for all** node $c \in \mathcal{C}_{t_i}$ **do**

      $\mathcal{P}_c \leftarrow \{c\} \cup \bigcup_{c' \in pred^{+}_{G_{t_i}}(c)} path_{G_{t_i}}(c', c)$

      $\mathcal{R}_c \leftarrow \{R \subseteq \{0, \ldots cp(c)\} \times \mathcal{P}_c \mid \sum_{(k,\rho) \in R} k = cp(c)\}$

    **end for**

    $Tr(s) \leftarrow \{\bigcup_{c \in \mathcal{C}_{t_i}} R_c \mid \forall c \in \mathcal{C}_{t_i} : R_c \in \mathcal{R}_c\}$

    **for all** $R \in Tr(s)$ **do**

      $s' \leftarrow get\_previous\_state(s, R)$

      $\mathcal{S}_{t_i} \leftarrow \mathcal{S}_{t_i} \cup \{s'\}$

      $pr_R \leftarrow SDP(R, s', t_i)$

      **if** $Pr(s')$ is undefined or $Pr(s') < pr_R$ **then**

        $Pr(s') \leftarrow pr_R$

        $best\_action(s') \leftarrow R$

      **end if**

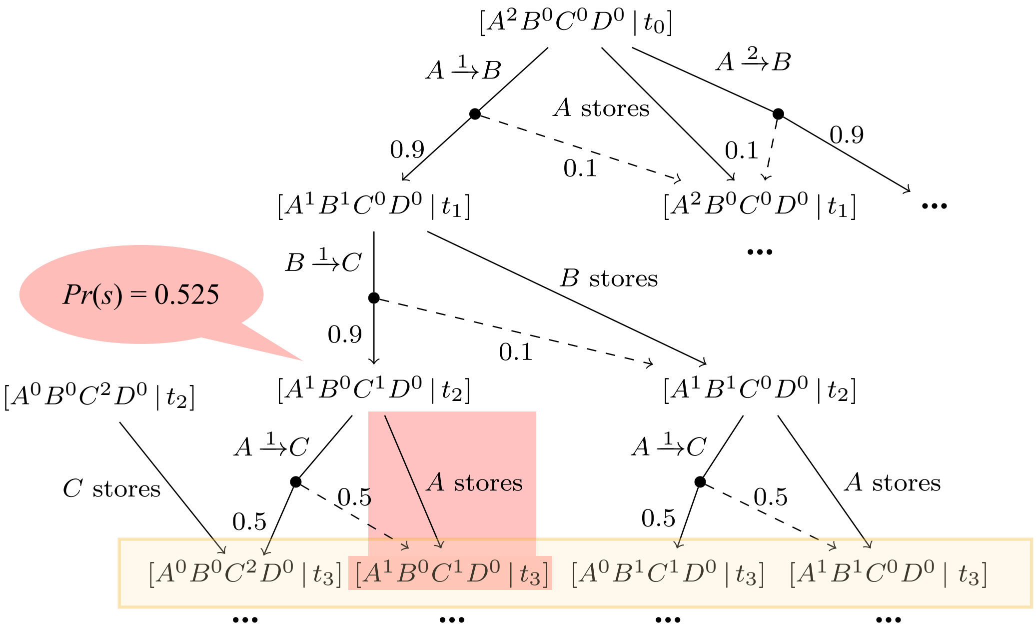
    **end for**

    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_{t_i}$

  **end for**

**end for**

**return** $\mathcal{S}$, $Tr$, $Pr$




$$\left\{ [A^w B^x C^y D^z \mid t_4] \;\middle|\; z > 1 \wedge w + x + y + z = 2 \right\}$$

Raverta et al., 2021. Ad Hoc Networks

UNC

## First technique

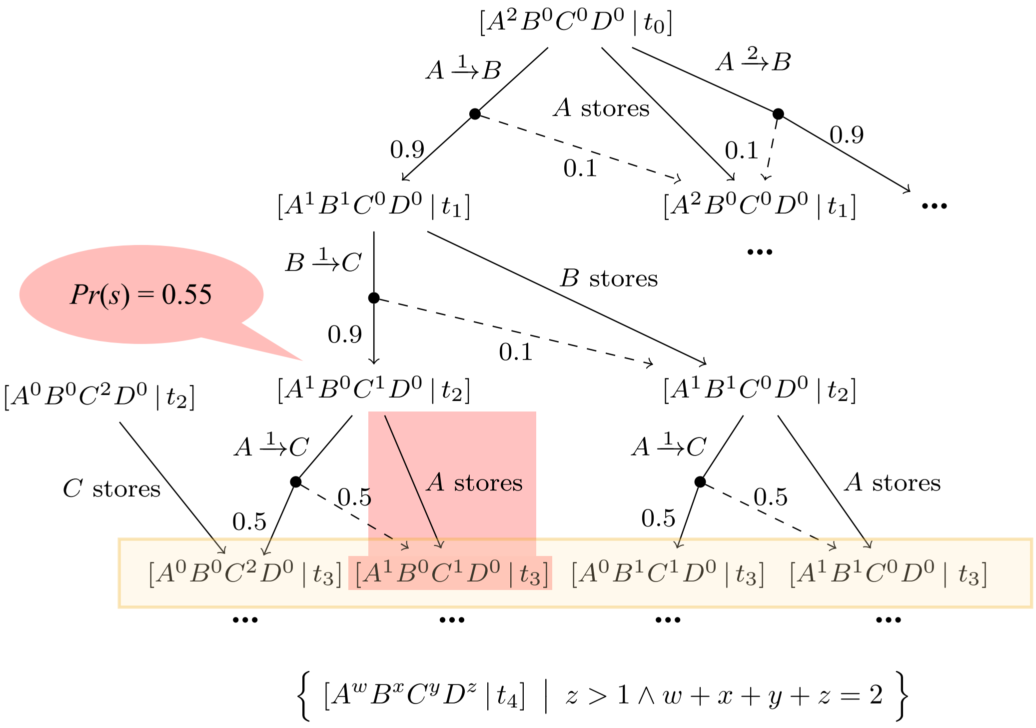
# Routing under Uncertain Contact Plans (RUCoP)

**Algorithm 1:** The RUCoP algorithm

**Input:** Uncertain time varying graph $\mathcal{G}$, $num\_copies$, Target

**Output:** Explored states $\mathcal{S}$, Routing table $Tr$, Successful delivery probability $Pr$

determine *successful states* $\mathcal{S}_{t_{end}}$ for $num\_copies$
$\mathcal{S} \leftarrow \mathcal{S}_{t_{end}}$
**for all** $t_i \in \mathcal{T}$, starting from $t_{end-1}$ **do**
  $\mathcal{S}_{t_i} \leftarrow \varnothing$
  **for all** state $s \in \mathcal{S}_{t_{i+1}}$ **do**
    determine *carrier nodes* $\mathcal{C}_{t_i}$
    **for all** node $c \in \mathcal{C}_{t_i}$ **do**
      $\mathcal{P}_c \leftarrow \{c\} \cup \bigcup_{c' \in pred^+_{G_{t_i}}(c)} path_{G_{t_i}}(c', c)$
      $\mathcal{R}_c \leftarrow \left\{ R \subseteq \{0, \ldots cp(c)\} \times \mathcal{P}_c \mid \sum_{(k,\rho)\in R} k = cp(c) \right\}$
    **end for**
    $Tr(s) \leftarrow \left\{ \bigcup_{c \in \mathcal{C}_{t_i}} R_c \mid \forall c \in \mathcal{C}_{t_i} : R_c \in \mathcal{R}_c \right\}$
    **for all** $R \in Tr(s)$ **do**
      $s' \leftarrow get\_previous\_state(s, R)$
      $\mathcal{S}_{t_i} \leftarrow \mathcal{S}_{t_i} \cup \{s'\}$
      $pr_R \leftarrow SDP(R, s', t_i)$
      **if** $Pr(s')$ is undefined or $Pr(s') < pr_R$ **then**
        $Pr(s') \leftarrow pr_R$
        $best\_action(s') \leftarrow R$
      **end if**
    **end for**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_{t_i}$
  **end for**
**end for**
**return** $\mathcal{S}$, $Tr$, $Pr$






UNC

First technique

# Routing under Uncertain Contact Plans (RUCoP)

**Algorithm 1:** The RUCoP algorithm

**Input:** Uncertain time varying graph $\mathcal{G}$, $num\_copies$, Target

**Output:** Explored states $\mathcal{S}$, Routing table $Tr$, Successful delivery probability $Pr$

determine *successful states* $\mathcal{S}_{t_{end}}$ for $num\_copies$

$\mathcal{S} \leftarrow \mathcal{S}_{t_{end}}$

**for all** $t_i \in \mathcal{T}$, starting from $t_{end-1}$ **do**

  $\mathcal{S}_{t_i} \leftarrow \varnothing$

  **for all** state $s \in \mathcal{S}_{t_{i+1}}$ **do**

    determine *carrier nodes* $\mathcal{C}_{t_i}$

    **for all** node $c \in \mathcal{C}_{t_i}$ **do**

      $\mathcal{P}_c \leftarrow \{c\} \cup \bigcup_{c' \in pred^+_{G_{t_i}}(c)} path_{G_{t_i}}(c', c)$

      $\mathcal{R}_c \leftarrow \left\{ R \subseteq \{0, \ldots cp(c)\} \times \mathcal{P}_c \mid \sum_{(k,\rho) \in R} k = cp(c) \right\}$

    **end for**

    $Tr(s) \leftarrow \left\{ \bigcup_{c \in \mathcal{C}_{t_i}} R_c \mid \forall c \in \mathcal{C}_{t_i} : R_c \in \mathcal{R}_c \right\}$

    **for all** $R \in Tr(s)$ **do**

      $s' \leftarrow get\_previous\_state(s, R)$

      $\mathcal{S}_{t_i} \leftarrow \mathcal{S}_{t_i} \cup \{s'\}$

      $pr_R \leftarrow SDP(R, s', t_i)$

      **if** $Pr(s')$ is undefined or $Pr(s') < pr_R$ **then**

        $Pr(s') \leftarrow pr_R$

        $best\_action(s') \leftarrow R$

      **end if**

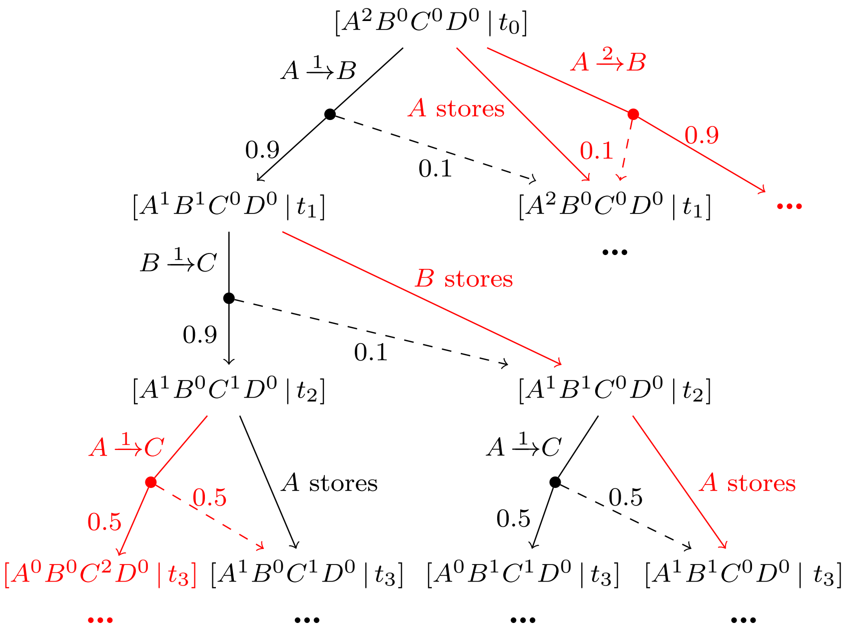
    **end for**

    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_{t_i}$

  **end for**

**end for**

**return** $\mathcal{S}$, $Tr$, $Pr$




$$\left\{ [A^w B^x C^y D^z \mid t_4] \;\middle|\; z > 1 \land w + x + y + z = 2 \right\}$$

Raverta et al., 2021. Ad Hoc Networks

Second technique

# Simulation through Lightweight Scheduler Sampling (LSS)

One simulation run

SMC+LSS:

1. Select $m$ 32-bit integer, each of them representing a scheduler identifier σ
2. For each σ, perform standard SMC letting σ resolve all non-determinism
3. Return the estimated value and the corresponding σ

**Input:**
Network of VMDP $M = \|_{SV}(M_1, \ldots, M_n)$ with $[\![M]\!] = \langle S, s_I, A, T\rangle$, goal set $G \subseteq S$, $\sigma \in \mathbb{Z}_{32}$, $\mathcal{H}$ uniform deterministic, PRNG $\mathcal{U}_{\text{pr}}$.

$s := s_I$
**while** $s \notin G$ **do** // *break on goal state*
  **if** $\forall\, s \xrightarrow{a} \mu\colon \mu = \{\, s \mapsto 1 \,\}$ **then break** // *break on self-loops*
  $\langle a, \mu\rangle := (\mathcal{H}(\sigma.s) \bmod |T\,|)$-th element of $T$ // *schedule transition*
  $s := \mathcal{U}_{\text{pr}}(\mu)$ // *select next state according to* $\mu$
**return** $s \in G$

CONICET

Legay, Sedwards, & Traounez 2014, SEFM // D'Argenio Legay, Sewards, & Traounez 2015, STTT

UNC

Second technique

# Simulation through Lightweight Scheduler Sampling (LSS)

SMC+LSS:

1. Select $m$ 32-bit integer, each of them representing a scheduler identifier σ
2. For each σ, perform standard SMC letting σ resolve all non-determinism
3. Return the estimated value and the corresponding σ

One simulation run

**Input:**
Network of VMDP $M = \|_{SV}(M_1, \ldots, M_n)$ with $[\![M]\!] = \langle S, s_I, A, T\rangle$, goal set $G \subseteq S$, $\sigma \in \mathbb{Z}_{32}$, $\mathcal{H}$ uniform deterministic, PRNG $\mathcal{U}_{\mathrm{pr}}$.

$s := s_I$
**while** $s \notin G$ **do** // *break on goal state*
  **if** $\forall\, s \xrightarrow{a} \mu\colon \mu = \{\, s \mapsto 1 \,\}$ **then break** // *break on self-loops*
  $\langle a, \mu\rangle := (\mathcal{H}(\sigma.s) \bmod |T\,|)$-th element of $T$ // *schedule transition*
  $s := \mathcal{U}_{\mathrm{pr}}(\mu)$ // *select next state according to* $\mu$
**return** $s \in G$

Second technique

# Simulation through Lightweight Scheduler Sampling (LSS)

One simulation run

SMC+LSS:

1. Select $m$ 32-bit integer, each of them representing a scheduler identifier σ
2. For each σ, perform standard SMC letting σ resolve all non-determinism
3. Return the estimated value and the corresponding σ

**Input:**
Network of VMDP $M = \|_{SV}(M_1, \ldots, M_n)$ with $[\![M]\!] = \langle S, s_I, A, T \rangle$, goal set $G \subseteq S$, $\sigma \in \mathbb{Z}_{32}$, $\mathcal{H}$ uniform deterministic, PRNG $\mathcal{U}_{\text{pr}}$.

$s := s_I$
**while** $s \notin G$ **do** // *break on goal state*
  **if** $\forall\, s \xrightarrow{a} \mu \colon \mu = \{\, s \mapsto 1 \,\}$ **then break** // *break on self-loops*
  $\langle a, \mu \rangle := (\mathcal{H}(\sigma.s) \bmod |T\,|)$-th element of $T$ // *schedule transition*
  $s := \mathcal{U}_{\text{pr}}(\mu)$ // *select next state according to* $\mu$
**return** $s \in G$

Second technique

# Simulation through Lightweight Scheduler Sampling (LSS)

One simulation run

SMC+LSS:

1. Select $m$ 32-bit integer, each of them representing a scheduler identifier σ
2. For each σ, perform standard SMC letting σ resolve all non-determinism
3. Return the estimated value and the corresponding σ

**Input:**
Network of VMDP $M = \|_{SV}(M_1, \ldots, M_n)$ with $[\![M]\!] = \langle S, s_I, A, T \rangle$, goal set $G \subseteq S$, $\sigma \in \mathbb{Z}_{32}$, $\mathcal{H}$ uniform deterministic, PRNG $\mathcal{U}_{\mathrm{pr}}$.

$s := s_I$
**while** $s \notin G$ **do** *// break on goal state*
  **if** $\forall s \xrightarrow{a} \mu : \mu = \{ s \mapsto 1 \}$ **then break** *// break on self-loops*
  $\langle a, \mu \rangle := (\mathcal{H}(\sigma.s) \bmod |T|)$-th element of $T$ *// schedule transition*
  $s := \mathcal{U}_{\mathrm{pr}}(\mu)$ *// select next state according to $\mu$*
**return** $s \in G$

Hash key obtained by concatenating the scheduler with the state

The hash function returns a 32-bit number which is used to select the transition

Second technique

# Simulation through Lightweight Scheduler Sampling (LSS)

SMC+LSS:

1. Select $m$ 32-bit integer, each of them representing a scheduler identifier σ
2. For each σ, perform standard SMC letting σ resolve all non-determinism
3. Return the estimated value and the corresponding σ

One simulation run

**Input:**
Network of VMDP $M = \|_{SV}(M_1, \ldots, M_n)$ with $[\![M]\!] = \langle S, s_I, A, T\rangle$, goal set $G \subseteq S$, $\sigma \in \mathbb{Z}_{32}$, $\mathcal{H}$ uniform deterministic, PRNG $\mathcal{U}_{\text{pr}}$.

$s := s_I$
**while** $s \notin G$ **do** // *break on goal state*
  **if** $\forall\, s \xrightarrow{a} \mu\colon \mu = \{\, s \mapsto 1 \,\}$ **then break** // *break on self-loops*
  $\langle a, \mu\rangle := (\mathcal{H}(\sigma.s) \bmod |T|)$-th element of $T$ // *schedule transition*
  $s := \mathcal{U}_{\text{pr}}(\mu)$ // *select next state according to* $\mu$
**return** $s \in G$

Second technique

# Simulation through Lightweight Scheduler Sampling (LSS)

SMC+LSS:

1. Select $m$ 32-bit integer, each of them representing a scheduler identifier σ
2. For each σ, perform standard SMC letting σ resolve all non-determinism
3. Return the estimated value and the corresponding σ

One simulation run

**Input:**
Network of VMDP $M = \|_{SV}(M_1, \ldots, M_n)$ with $[\![M]\!] = \langle S, s_I, A, T \rangle$, goal set $G \subseteq S$, $\sigma \in \mathbb{Z}_{32}$, $\mathcal{H}$ uniform deterministic, PRNG $\mathcal{U}_{\text{pr}}$.

$s := s_I$
**while** $s \notin G$ **do** // *break on goal state*
  **if** $\forall\, s \xrightarrow{a} \mu \colon \mu = \{ s \mapsto 1 \}$ **then break** // *break on self-loops*
  $\langle a, \mu \rangle := (\mathcal{H}(\sigma.s) \bmod |T|)$-th element of $T$ // *schedule transition*
  $s := \mathcal{U}_{\text{pr}}(\mu)$ // *select next state according to* $\mu$
**return** $s \in G$

Second technique

# Simulation through Lightweight Scheduler Sampling (LSS)

SMC+LSS:

1. Select $m$ 32-bit integer, each of them representing a scheduler identifier σ
2. For each σ, perform standard SMC letting σ resolve all non-determinism
3. Return the estimated value and the corresponding σ

One simulation run

**Input:**
Network of VMDP $M = \|_{SV}(M_1, \ldots, M_n)$ with $[\![M]\!] = \langle S, s_I, A, T \rangle$, goal set $G \subseteq S$, $\sigma \in \mathbb{Z}_{32}$, $\mathcal{H}$ uniform deterministic, PRNG $\mathcal{U}_{\mathrm{pr}}$.

$s := s_I$
**while** $s \notin G$ **do** *// break on goal state*
  **if** $\forall s \xrightarrow{a} \mu \colon \mu = \{ s \mapsto 1 \}$ **then break** *// break on self-loops*
  $\langle a, \mu \rangle := (\mathcal{H}(\sigma.s) \bmod |T|)$-th element of $T$ *// schedule transition*
  $s := \mathcal{U}_{\mathrm{pr}}(\mu)$ *// select next state according to* $\mu$
**return** $s \in G$

Sim($\sigma$) estimates $\Diamond G$ by running this algorithm multiple times

Second technique

# Simulation through Lightweight Scheduler Sampling (LSS)

SMC+LSS:

1. Select $m$ 32-bit integer, each of them representing a scheduler identifier σ
2. For each σ, perform standard SMC letting σ resolve all non-determinism
3. Return the estimated value and the corresponding σ

One simulation run

**Input:**
Network of VMDP $M = \|_{SV}(M_1, \ldots, M_n)$ with $[\![M]\!] = \langle S, s_I, A, T \rangle$, goal set $G \subseteq S$, $\sigma \in \mathbb{Z}_{32}$, $\mathcal{H}$ uniform deterministic, PRNG $\mathcal{U}_{\text{pr}}$.

```
s := s_I
while s ∉ G do                                          // break on goal state
    if ∀ s -a-> μ : μ = { s ↦ 1 } then break            // break on self-loops
    ⟨a, μ⟩ := (H(σ.s) mod |T|)-th element of T          // schedule transition
    s := U_pr(μ)                                        // select next state according to μ
return s ∈ G
```

- SMC+LSS returns an underapproximation (or overapproximation) which we call near optimal
- It corresponds to the σ reporting the best (max or min) estimated value
- The efficiency depends on $m$

Sim($\sigma$) estimates $\diamond G$ by running this algorithm multiple times

$$\text{near\_max} = \max \{\text{Sim}(\sigma_i) \mid 1 \leq i \leq m\}$$

$$\text{near\_min} = \min \{\text{Sim}(\sigma_i) \mid 1 \leq i \leq m\}$$

with $\sigma_i \in \mathbb{Z}_{32}$ for all $1 \leq i \leq m$

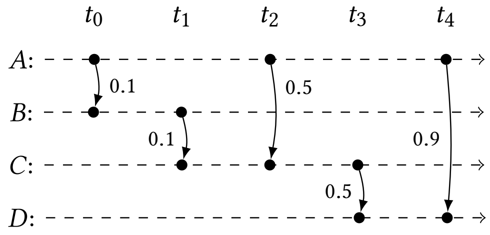
Second technique

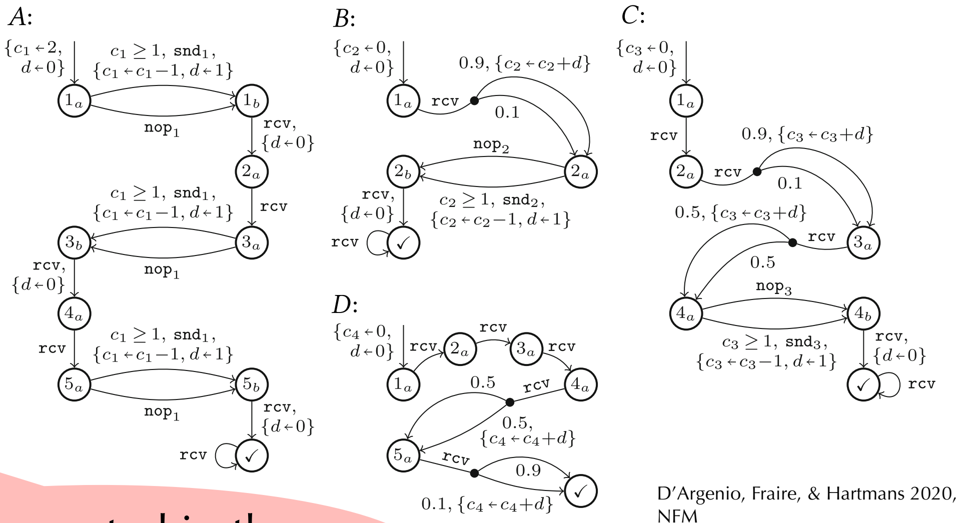
# Simulation through Lightweight Scheduler Sampling (LSS)

SMC+LSS:

1. Select $m$ 32-bit integer, each of them representing a scheduler identifier σ
2. For each σ, perform standard SMC letting σ resolve all non-determinism
3. Return the estimated value and the corresponding σ

- SMC+LSS returns an underapproximation (or overapproximation) which we call near optimal
- It corresponds to the σ reporting the best (max or min) estimated value
- The efficiency depends on $m$




D'Argenio, Fraire, & Hartmans 2020, NFM

Implemented in the MODEST toolset

Legay, Sedwards, & Traounez 2014, SEFM // D'Argenio Legay, Sewards, & Tra
Budde, D'Argenio, Hartmans, Sedwards 2020, STTT

Third technique

# Reinforcement Learning with Q-Learning

Objective: learn a matrix $Q\colon S \times Act \to [0,1]$ so that $\displaystyle\arg\max_{\langle a',\mu'\rangle \in Act(s)} Q(s,\langle a',\mu'\rangle)$ is the optimal choice

$$
\begin{array}{ll}
\textbf{for } i := 1 \textbf{ to } nr\_episodes \textbf{ do} & \\
\quad s := s_I & \\
\quad \textbf{while } s \notin G \textbf{ do} & \textit{// break on goal state} \\
\quad\quad \textbf{if } \forall\, s \xrightarrow{a} \mu\colon \mu = \{\, s \mapsto 1 \,\} \textbf{ then break} & \textit{// break on self-loops} \\
\quad\quad \langle a,\mu\rangle := \text{sample uniformly from } Act(s) & \\
\quad\quad\quad\quad \oplus_{\epsilon_i} \displaystyle\arg\max_{\langle a',\mu'\rangle \in Act(s)} Q(s,\langle a',\mu'\rangle) & \textit{// choose with probability } \epsilon_i \\
\quad\quad s' := \mathcal{U}_{\mathrm{pr}}(\mu) & \textit{// select next state according to } \mu \\
\quad\quad Q(s,\langle a,\mu\rangle) := (1-\alpha_i)\cdot Q(s,\langle a,\mu\rangle) & \\
\quad\quad\quad\quad + \alpha_i \cdot (\mathrm{Rew}(s') + \gamma \cdot \displaystyle\max_{\langle a',\mu'\rangle \in Act(s')} Q(s',\langle a',\mu'\rangle)) & \textit{// update } Q \textit{ matrix} \\
\quad\quad s := s' & \textit{// set new current state}
\end{array}
$$



Watkins & Peter Dayan, 1992. Mach. Learn.

Third technique

# Reinforcement Learning with Q-Learning

Objective: learn a matrix $Q\colon S \times Act \to [0,1]$ so that $\arg\max_{\langle a',\mu'\rangle \in Act(s)} Q(s,\langle a',\mu'\rangle)$ is the optimal choice

for all $i \geq 1$, $\epsilon_i > \epsilon_{i+1}$ and $\alpha_i > \alpha_{i+1}$

some conditions guarantee that converges to optimal as $nr\_episodes \to \infty$

```
for i := 1 to nr_episodes do
    s := s_I
    while s ∉ G do                                                // break on goal state
        if ∀ s -a-> μ: μ = { s ↦ 1 } then break                   // break on self-loops
        ⟨a, μ⟩ := sample uniformly from Act(s)
                  ⊕_{ε_i} argmax_{⟨a',μ'⟩∈Act(s)} Q(s, ⟨a', μ'⟩)   // choose with probability ε_i
        s' := U_pr(μ)                                             // select next state according to μ
        Q(s, ⟨a, μ⟩) := (1 − α_i) · Q(s, ⟨a, μ⟩)
                        + α_i · (Rew(s') + γ · max_{⟨a',μ'⟩∈Act(s')} Q(s', ⟨a', μ'⟩))   // update Q matrix
        s := s'                                                   // set new current state
```

CONICET

Watkins & Peter Dayan, 1992. Mach. Learn.

UNC

Third technique

# Reinforcement Learning with Q-Learning

Objective: learn a matrix $Q\colon S \times Act \to [0,1]$ so that $\arg\max_{\langle a',\mu'\rangle \in Act(s)} Q(s,\langle a',\mu'\rangle)$ is the optimal choice

for all $i \geq 1$, $\epsilon_i > \epsilon_{i+1}$ and $\alpha_i > \alpha_{i+1}$

some conditions guarantee that converges to optimal as $nr\_episodes \to \infty$

**for** $i := 1$ **to** $nr\_episodes$ **do**
  $s := s_I$
  **while** $s \notin G$ **do** — *// break on goal state*
    **if** $\forall\, s \xrightarrow{a} \mu\colon \mu = \{\, s \mapsto 1 \,\}$ **then break** — *// break on self-loops*
    $\langle a, \mu\rangle :=$ sample uniformly from $Act(s)$
      $\oplus_{\epsilon_i} \arg\max_{\langle a',\mu'\rangle \in Act(s)} Q(s,\langle a',\mu'\rangle)$ — *// choose with probability $\epsilon_i$*
    $s' := \mathcal{U}_{\mathrm{pr}}(\mu)$ — *// select next state according to $\mu$*
    $Q(s,\langle a,\mu\rangle) := (1-\alpha_i)\cdot Q(s,\langle a,\mu\rangle)$
      $+\, \alpha_i \cdot (\mathrm{Rew}(s') + \gamma \cdot \max_{\langle a',\mu'\rangle \in Act(s')} Q(s',\langle a',\mu'\rangle))$ — *// update Q matrix*
    $s := s'$ — *// set new current state*

CONICET

Watkins & Peter Dayan, 1992. Mach. Learn.

UNC

Third technique

# Reinforcement Learning with Q-Learning

Objective: learn a matrix $Q\colon S \times Act \to [0,1]$ so that $\displaystyle\arg\max_{\langle a',\mu'\rangle \in Act(s)} Q(s,\langle a',\mu'\rangle)$ is the optimal choice

for all $i \geq 1$, $\epsilon_i > \epsilon_{i+1}$ and $\alpha_i > \alpha_{i+1}$

some conditions guarantee that converges to optimal as $nr\_episodes \to \infty$

**for** $i := 1$ **to** $nr\_episodes$ **do**

- $s := s_I$
- **while** $s \notin G$ **do** — *// break on goal state*
  - **if** $\forall\, s \xrightarrow{a} \mu\colon \mu = \{\, s \mapsto 1 \,\}$ **then break** — *// break on self-loops*
  - $\langle a,\mu\rangle :=$ sample uniformly from $Act(s)$ $\displaystyle\oplus_{\epsilon_i} \arg\max_{\langle a',\mu'\rangle \in Act(s)} Q(s,\langle a',\mu'\rangle)$ — *// choose with probability $\epsilon_i$*
  - $s' := \mathcal{U}_{\mathrm{pr}}(\mu)$ — *// select next state according to $\mu$*
  - $\displaystyle Q(s,\langle a,\mu\rangle) := (1-\alpha_i)\cdot Q(s,\langle a,\mu\rangle) + \alpha_i \cdot (\mathbf{1}_G(s') + \max_{\langle a',\mu'\rangle \in Act(s')} Q(s',\langle a',\mu'\rangle))$ — *// update $Q$ matrix*
  - $s := s'$ — *// set new current state*

CONICET

Watkins & Peter Dayan, 1992. Mach. Learn.

UNC

Third technique

# Reinforcement Learning with Q-Learning

Objective: learn a matrix $Q\colon S \times Act \to [0,1]$ so that $\arg\max_{\langle a',\mu'\rangle \in Act(s)} Q(s,\langle a',\mu'\rangle)$ is the optimal choice

for all $i \geq 1$, $\epsilon_i > \epsilon_{i+1}$ and $\alpha_i > \alpha_{i+1}$

some conditions guarantee that converges to optimal as $nr\_episodes \to \infty$

**for** $i := 1$ **to** $nr\_episodes$ **do**
  Episode$(s_I, \epsilon_i, \alpha_i)$

Episode$(s,\epsilon,\alpha)$

$\langle a,\mu\rangle :=$ sample uniformly from $Act(s)$ $\oplus_\epsilon \arg\max_{\langle a',\mu'\rangle \in Act(s)} Q(s,\langle a',\mu'\rangle)$ // *choose with probability* $\epsilon$

$s' := \mathcal{U}_{\mathrm{pr}}(\mu)$ // *select next state according to* $\mu$

**if** $\forall\, s \xrightarrow{a} \mu \colon \mu = \{ s \mapsto 1 \}$ **then return** // *run ended unsuccessfully*

**else if** $s \in G$ **then return** // *run reached the goal*

**else** Episode$(s',\epsilon,\alpha)$

$$Q(s,\langle a,\mu\rangle) := (1-\alpha)\cdot Q(s,\langle a,\mu\rangle) + \alpha \cdot \left(\mathbf{1}_G(s') + \max_{\langle a',\mu'\rangle \in Act(s')} Q(s',\langle a',\mu'\rangle)\right)$$ // *update Q matrix*

Implemented in the MODEST toolset

CONICET

UNC

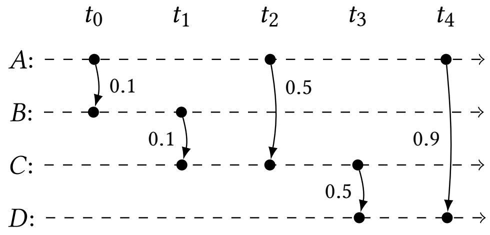
## Third technique

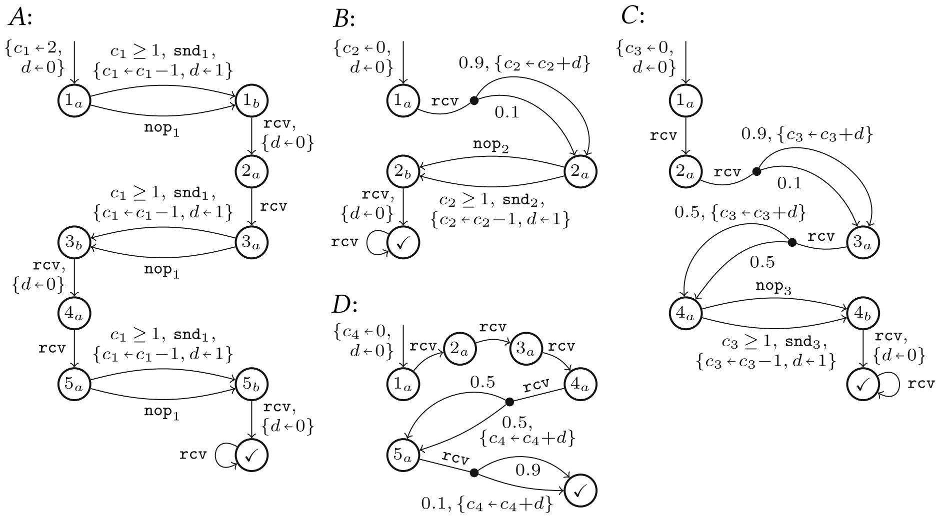
# Reinforcement Learning with Q-Learning

One simulation run

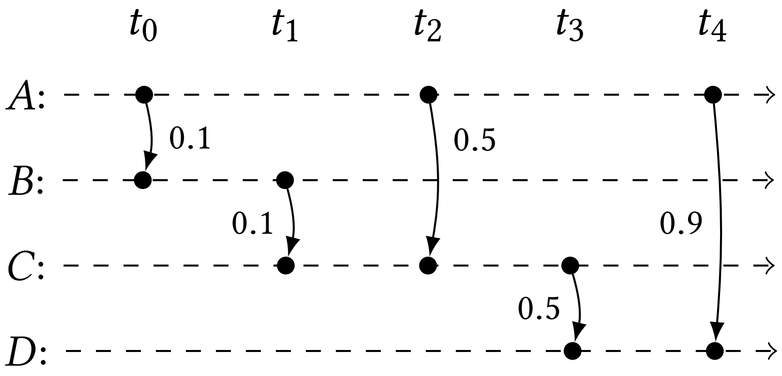
$s := s_I$
**while** $s \notin G$ **do** — *// break on goal state*
  **if** $\forall\, s \xrightarrow{a} \mu\colon \mu = \{\, s \mapsto 1 \,\}$ **then break** — *// break on self-loops*
  $\langle a, \mu \rangle := \arg\max_{\langle a', \mu' \rangle \in Act(s)} Q(s, \langle a', \mu' \rangle)$ — *// schedule transition*
  $s := \mathcal{U}_{\mathrm{pr}}(\mu)$ — *// select next state according to $\mu$*
**return** $s \in G$

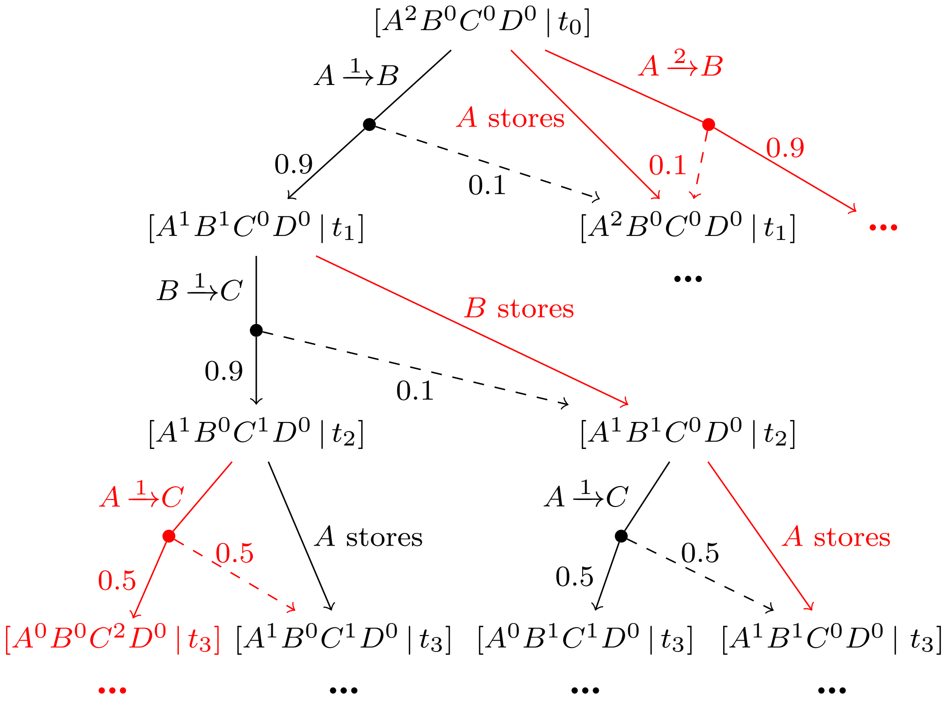
Implemented in the MODEST toolset



# The problem of distributed information

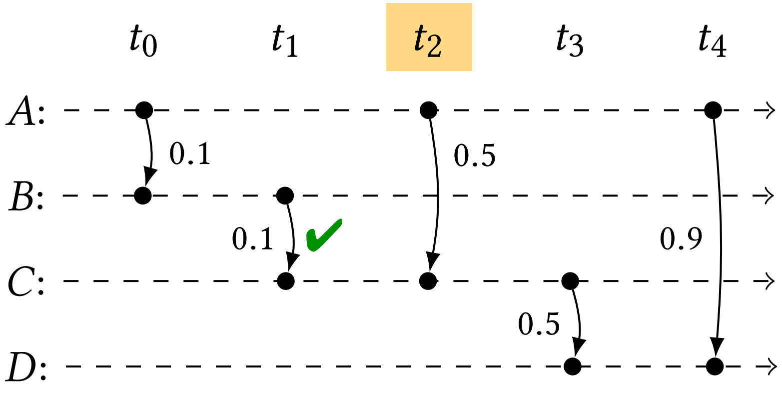


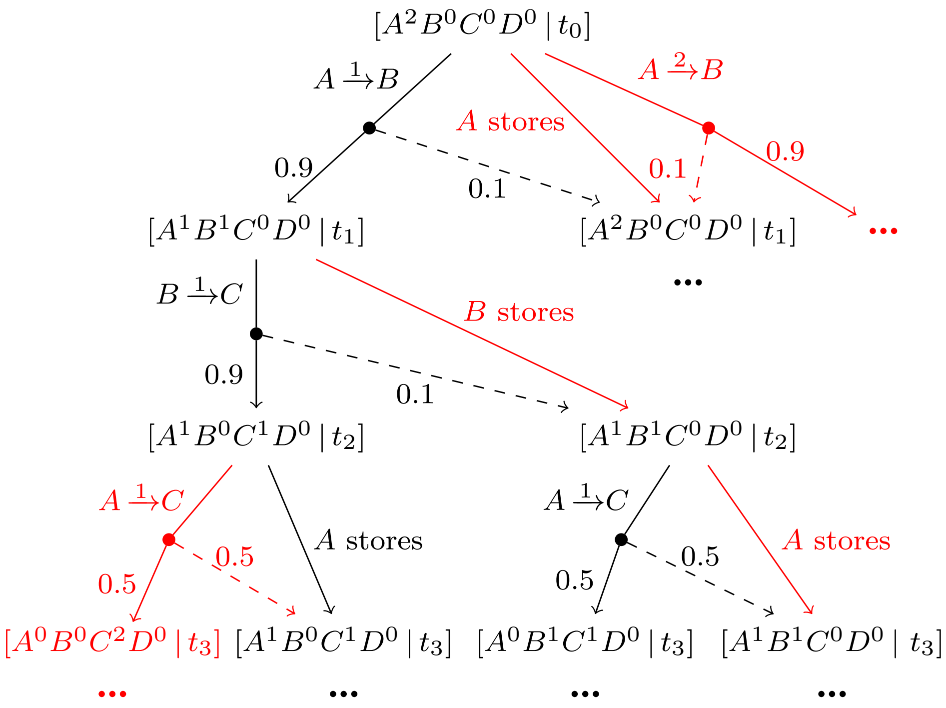




CONICET

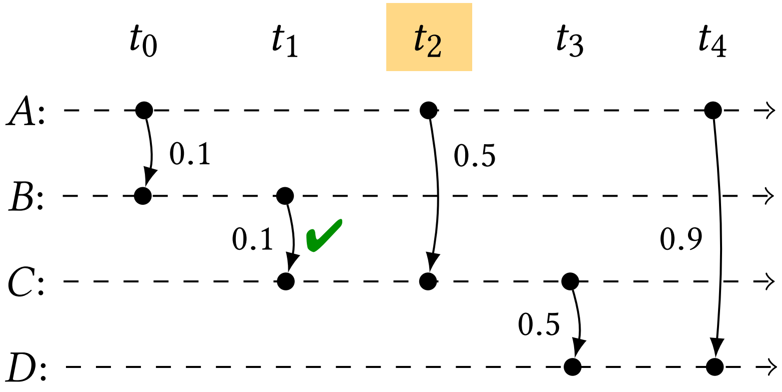
UNC

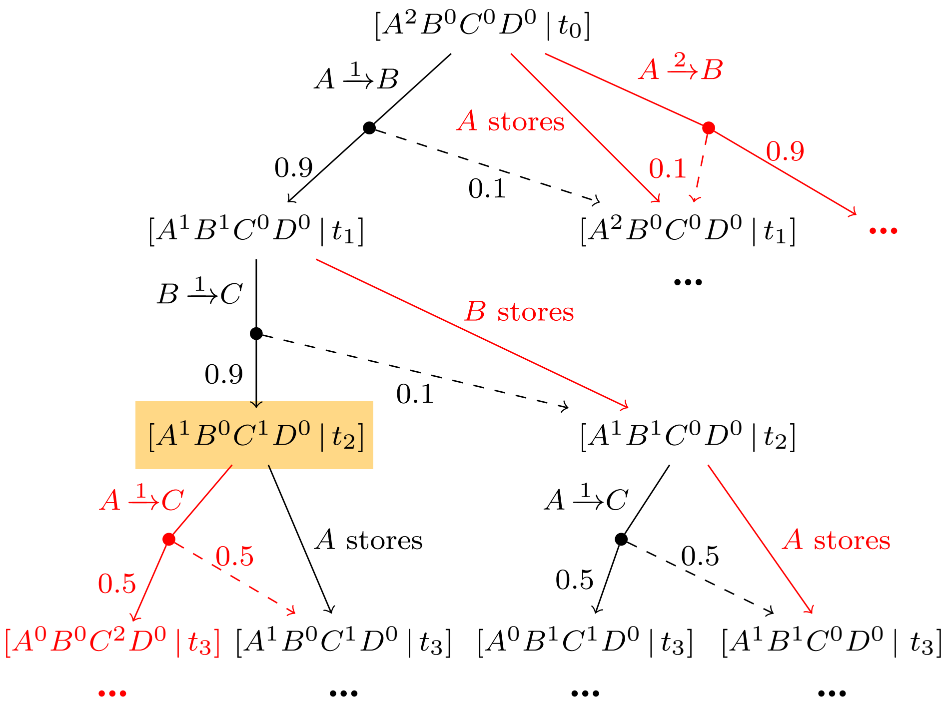
# The problem of distributed information



# The problem of distributed information

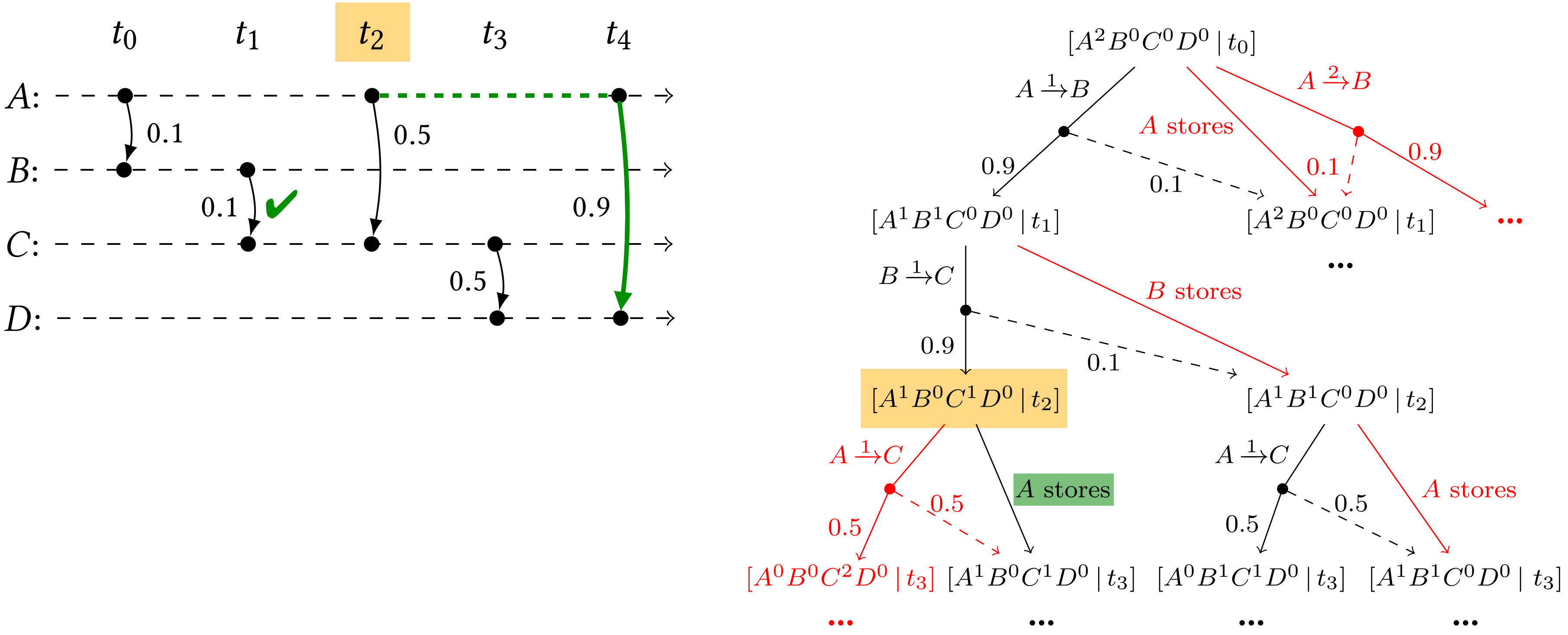


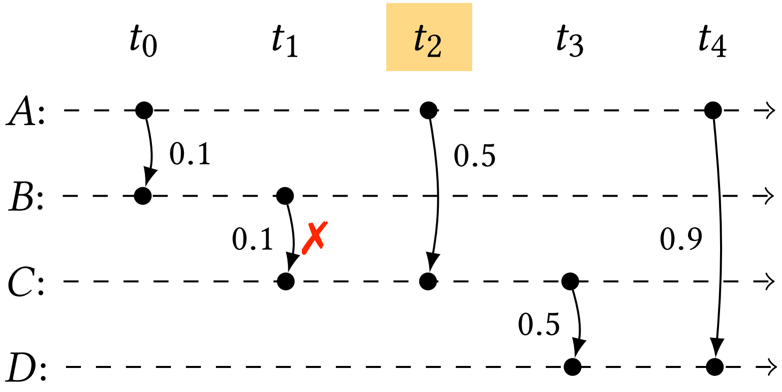

CONICET

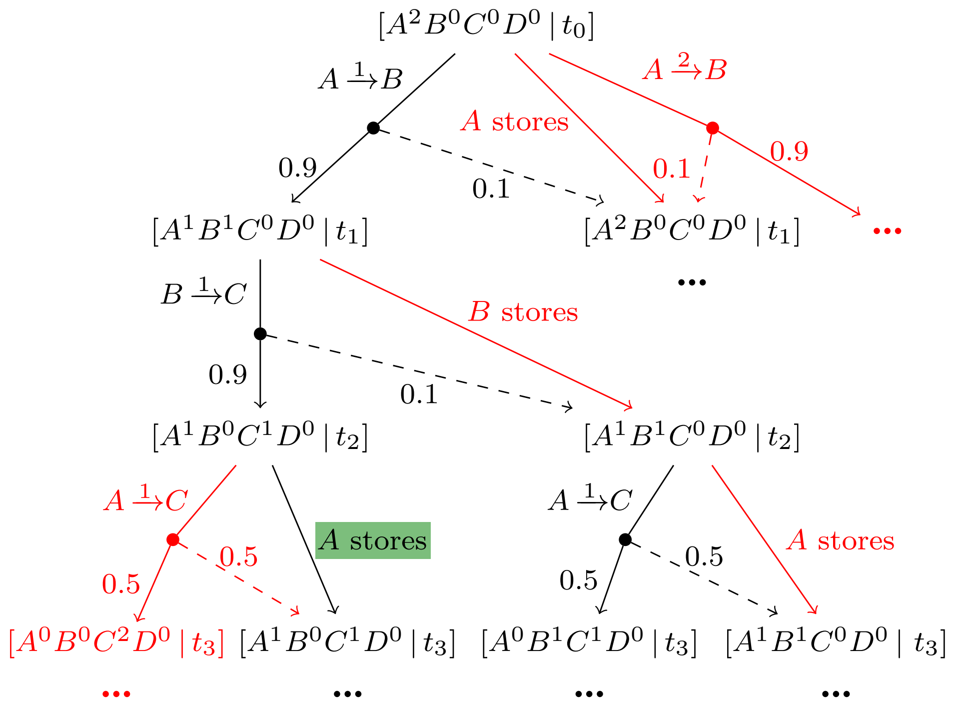
UNC

# The problem of distributed information



# The problem of distributed information

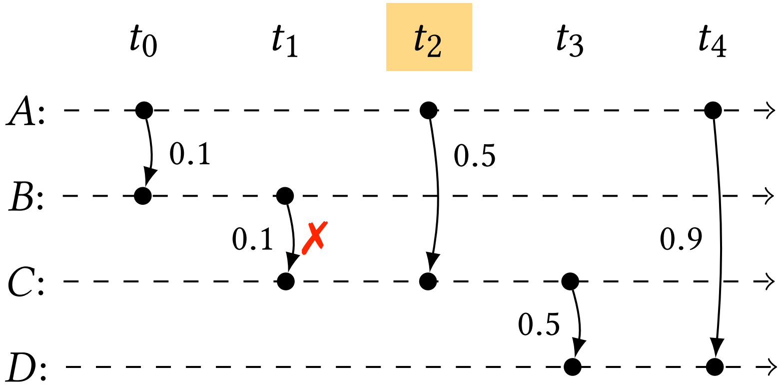


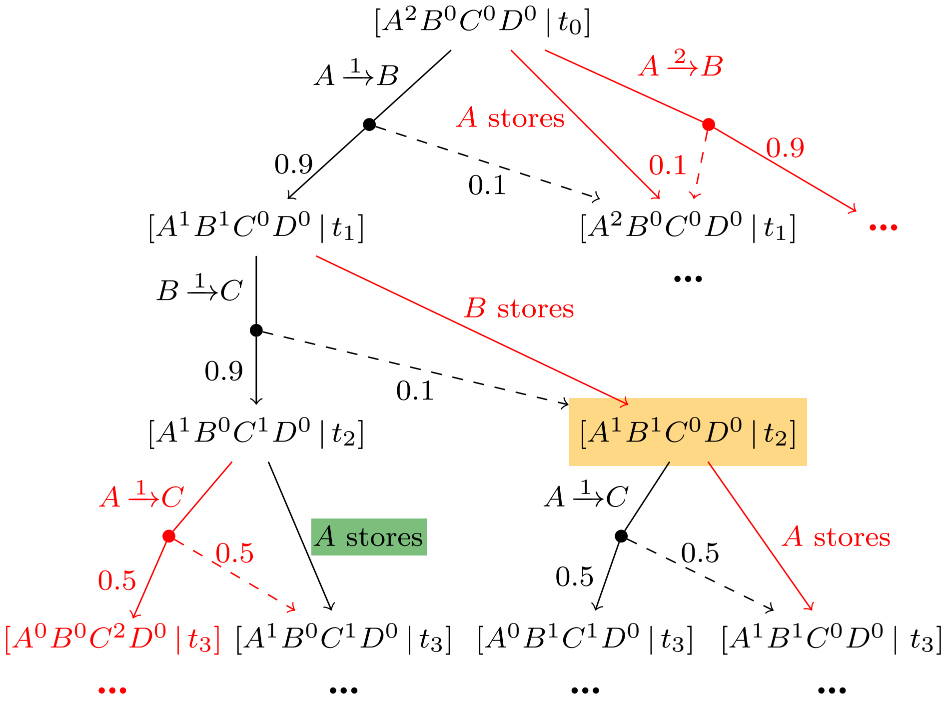




CONICET

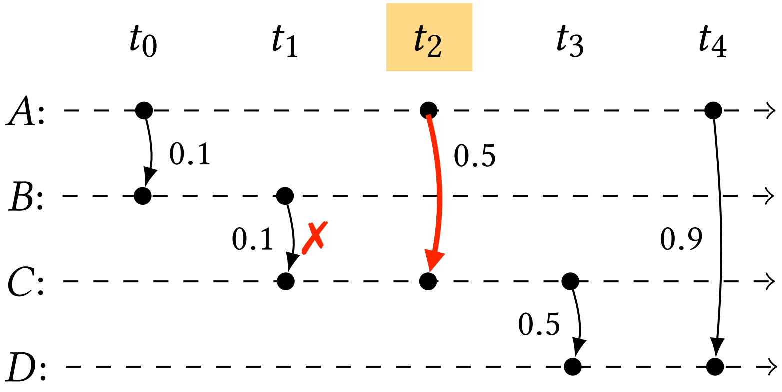
UNC

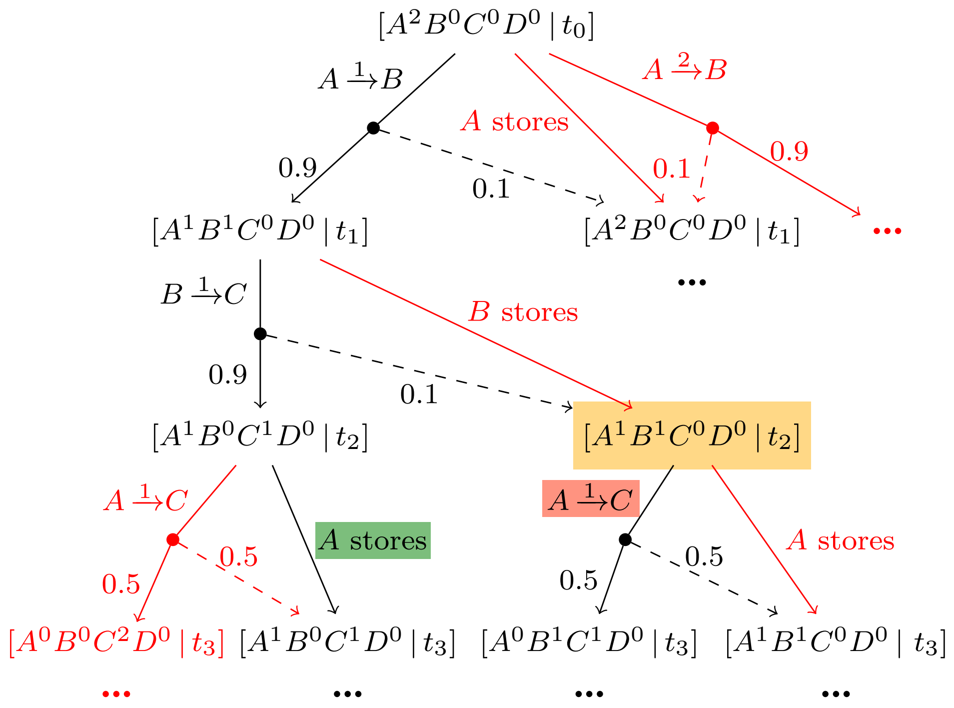
# The problem of distributed information



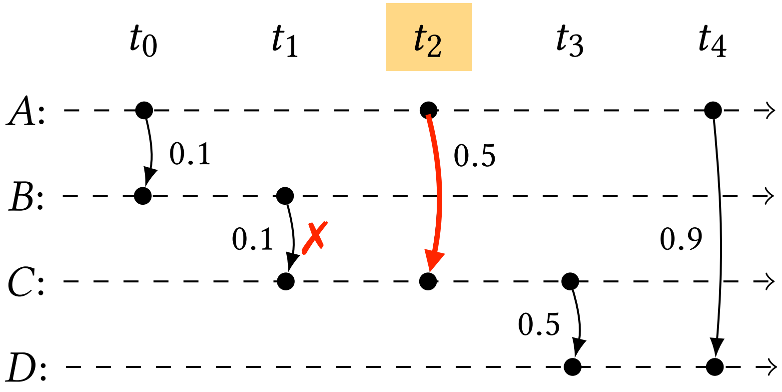



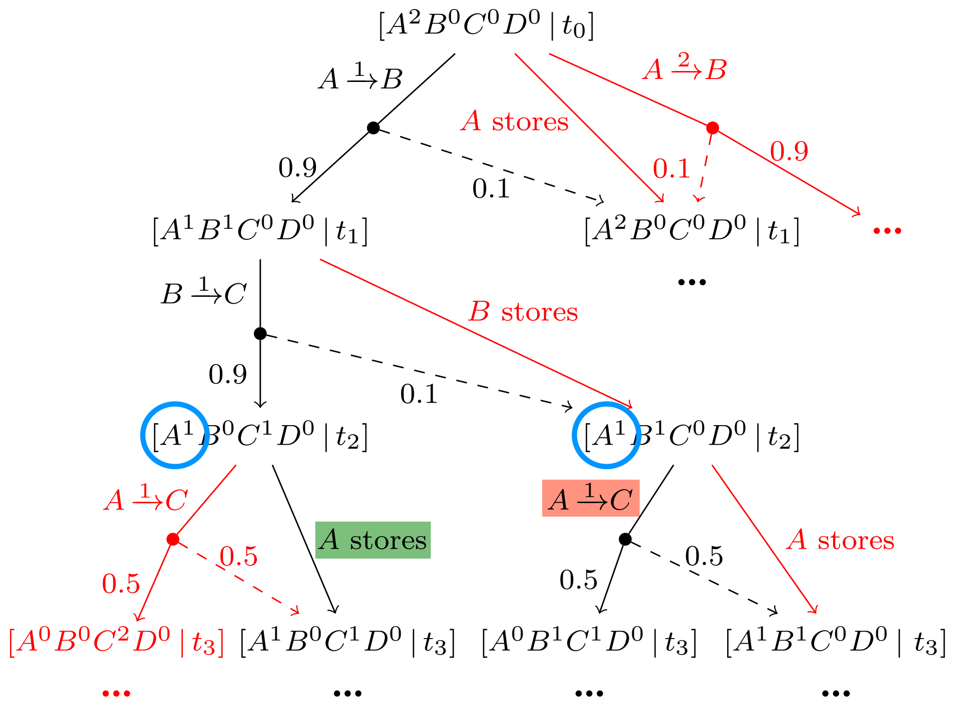
# The problem of distributed information



# The problem of distributed information

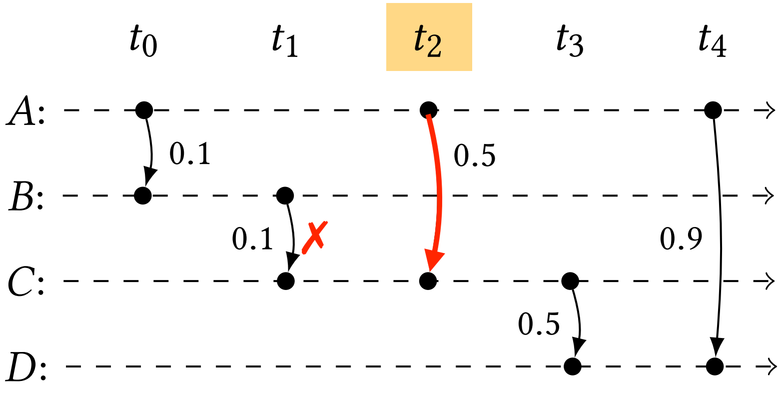


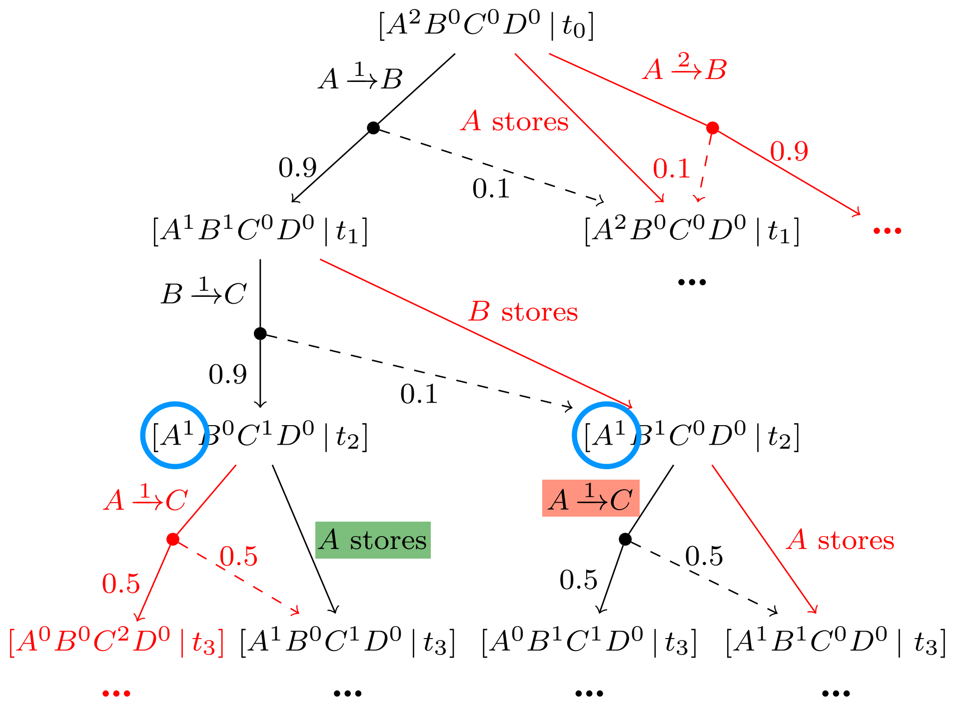

CONICET

UNC

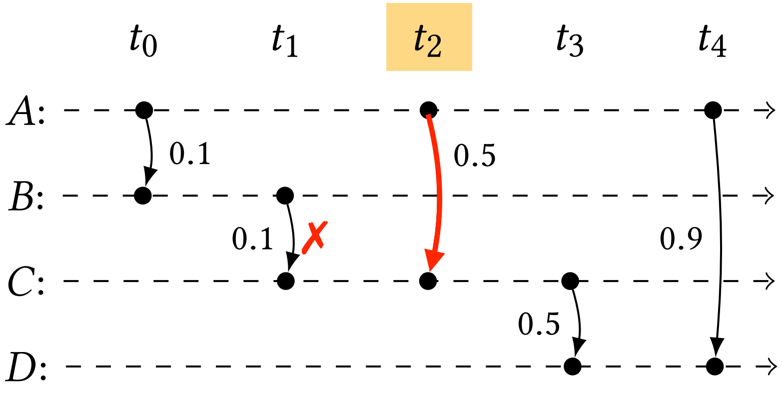
# The problem of distributed information

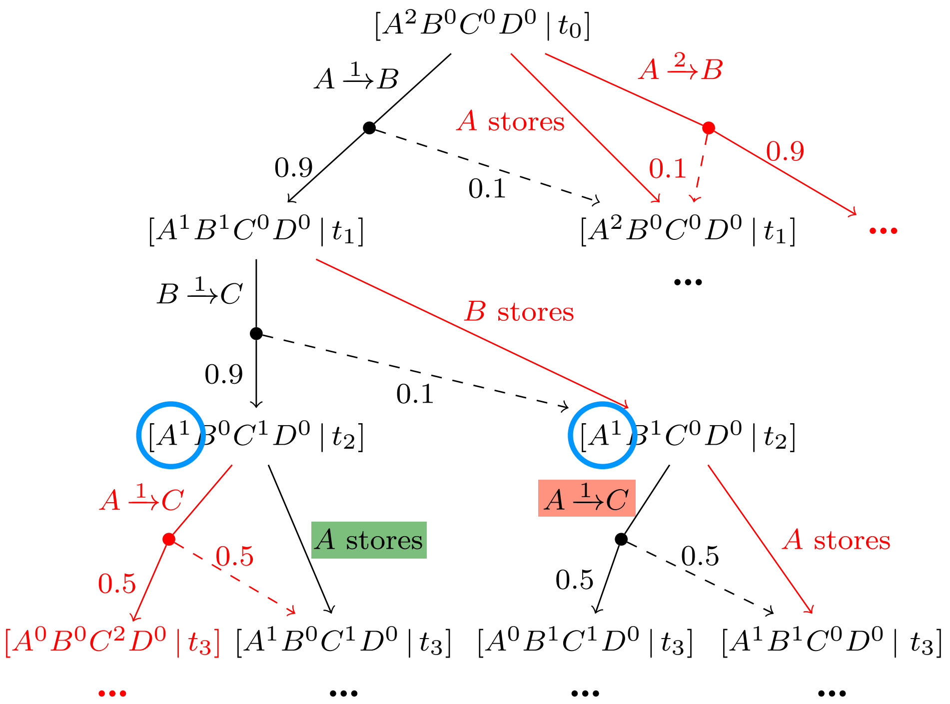



The decision has to be the same regardless the occurrences of locally unknown events

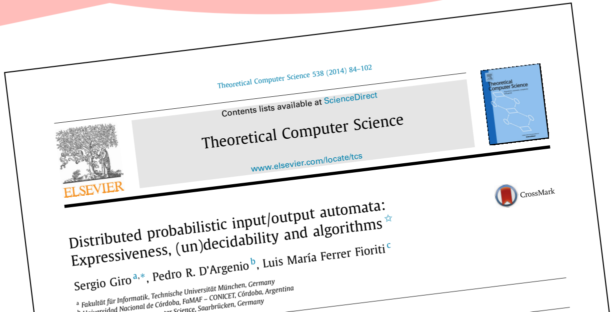


# The problem of distributed information



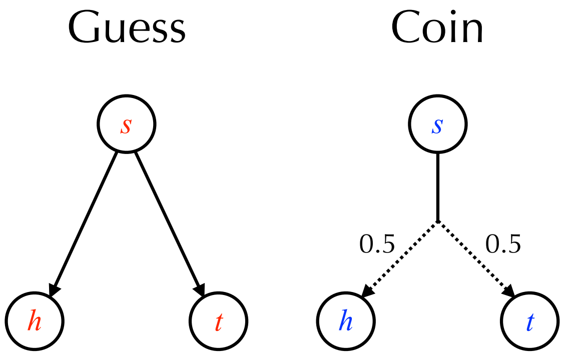



Theoretical Computer Science 538 (2014) 84–102
Contents lists available at ScienceDirect
Theoretical Computer Science
www.elsevier.com/locate/tcs
ELSEVIER

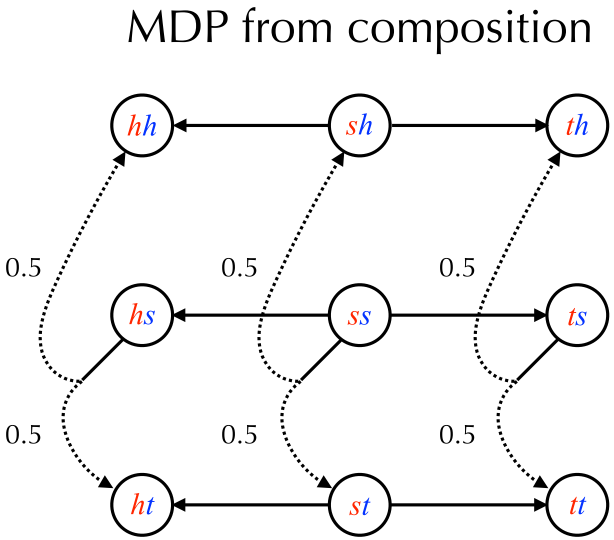
Distributed probabilistic input/output automata: Expressiveness, (un)decidability and algorithms ☆

Sergio Giro [a,*], Pedro R. D'Argenio [b], Luis María Ferrer Fioriti [c]
[a] Fakultät für Informatik, Technische Universität München, Germany
[b] Universidad Nacional de Córdoba, FaMAF – CONICET, Córdoba, Argentina

# Distributed schedulers

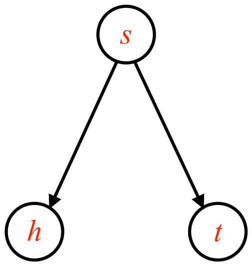



What is the probability of guessing?

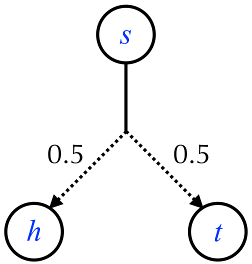


# Distributed schedulers

Guess

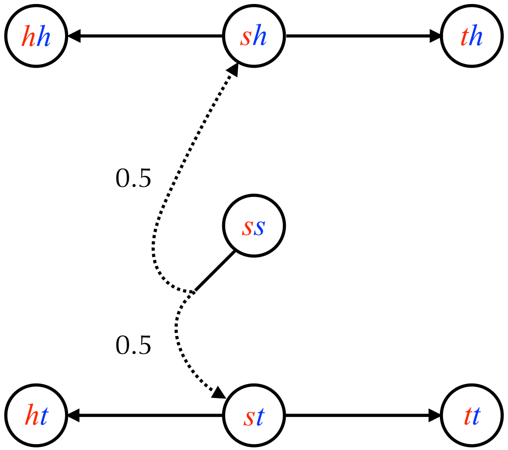



Coin




What is the probability of guessing?

MDP from composition

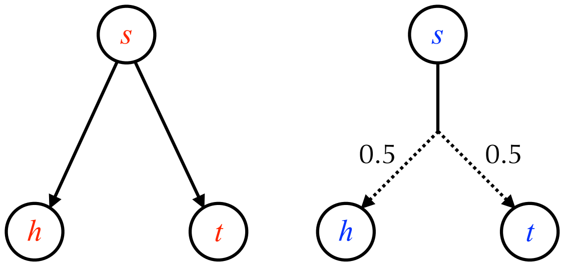



CONICET

UNC

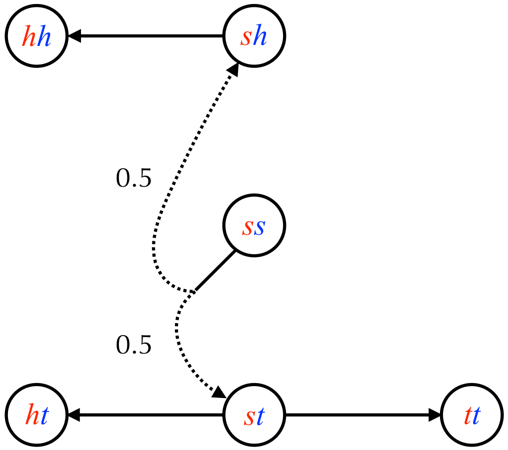
# Distributed schedulers

Guess    Coin



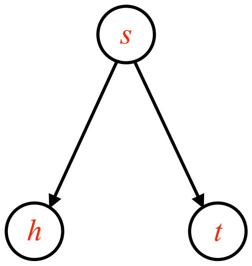

What is the probability of guessing?

MDP from composition

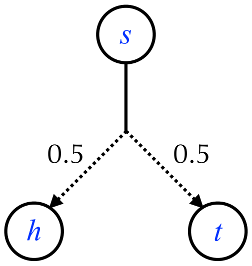


# Distributed schedulers

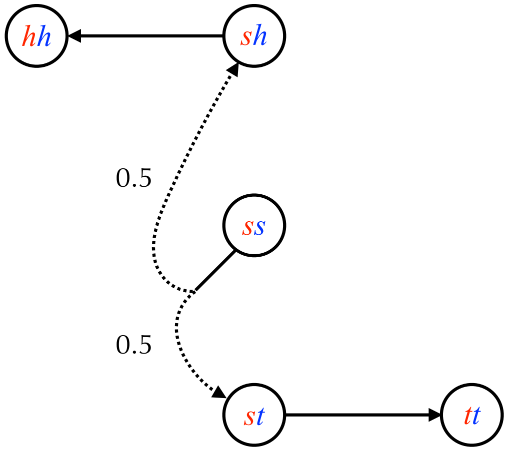
Guess




Coin




MDP from composition




What is the probability of guessing?

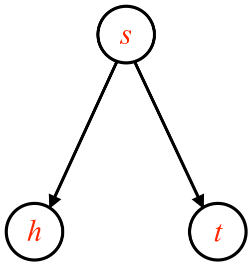
max prob of guessing = 1!

CONICET

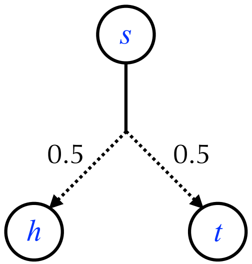
UNC

# Distributed schedulers

Guess

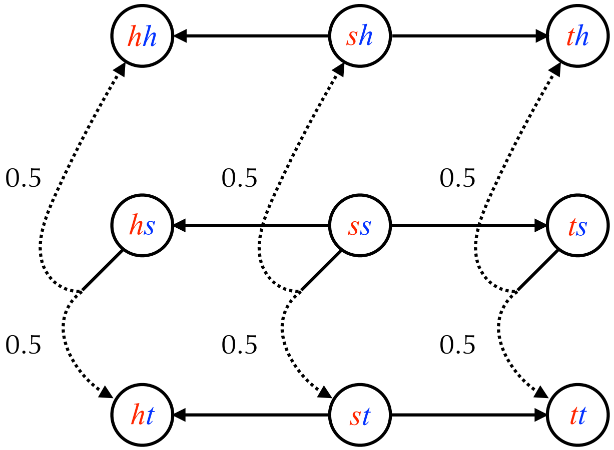



Coin



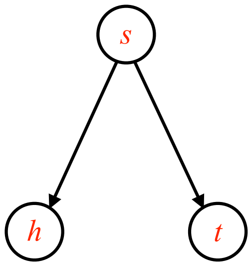

$$\mathfrak{S}(s)(a) = \sum_{i=1}^{n} \mathfrak{S}_c(s)(M_i) \cdot \mathfrak{S}_{M_i}(s \downarrow_{M_i})(a)$$

MDP from composition

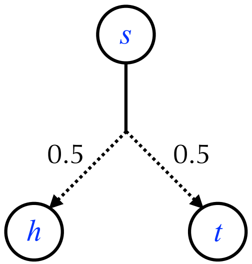


# Distributed schedulers

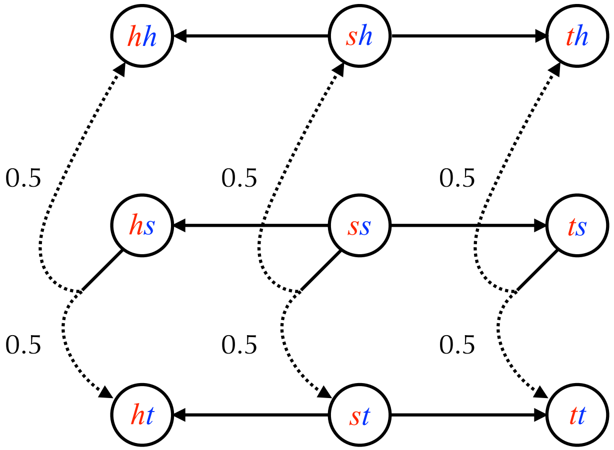
Guess




Coin




MDP from composition




$$\mathfrak{S}(s)(a) = \sum_{i=1}^{n} \mathfrak{S}_c(s)(M_i) \cdot \mathfrak{S}_{M_i}(s\downarrow_{M_i})(a)$$

global scheduler

component scheduler

local scheduler

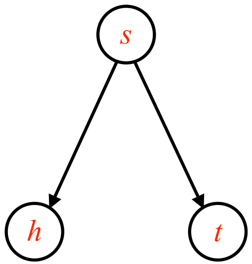
can only see the local projection

CONICET

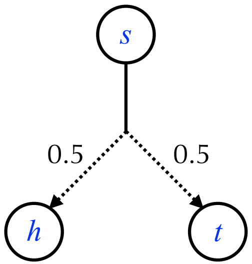
UNC

# Distributed schedulers

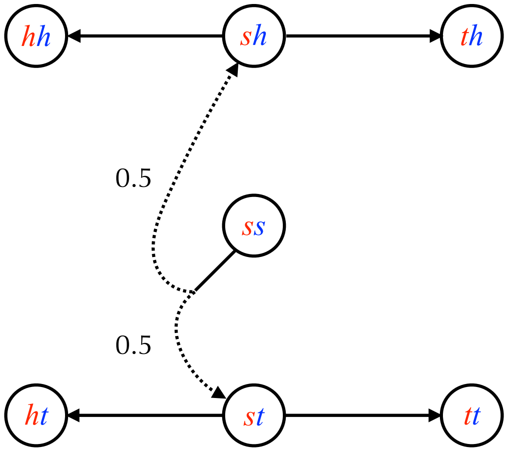
Guess




Coin




MDP from composition




$$\mathfrak{S}(s)(a) = \sum_{i=1}^{n} \mathfrak{S}_c(s)(M_i) \cdot \mathfrak{S}_{M_i}(s \downarrow_{M_i})(a)$$

global scheduler

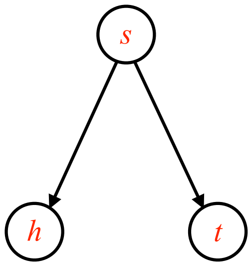
component scheduler

local scheduler

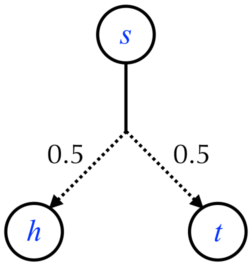
can only see the local projection

# Distributed schedulers

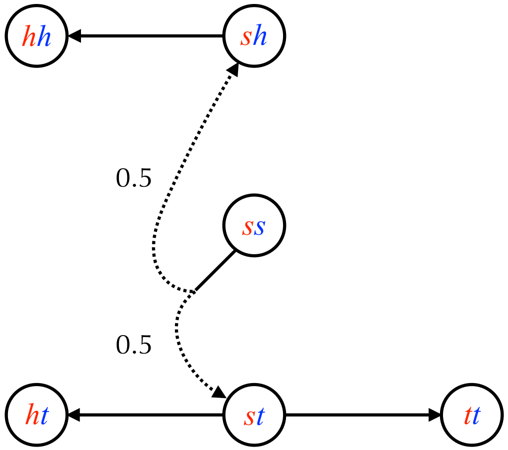
Guess




Coin




MDP from composition




$$\mathfrak{S}(s)(a) = \sum_{i=1}^{n} \mathfrak{S}_c(s)(M_i) \cdot \mathfrak{S}_{M_i}(s \downarrow_{M_i})(a)$$

global scheduler

component scheduler

local scheduler

can only see the local projection

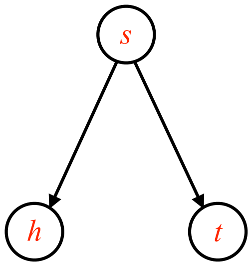
heads $= \mathfrak{S}_G(sh \downarrow_G)$

CONICET

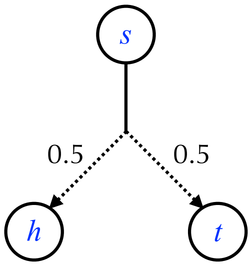
UNC

# Distributed schedulers

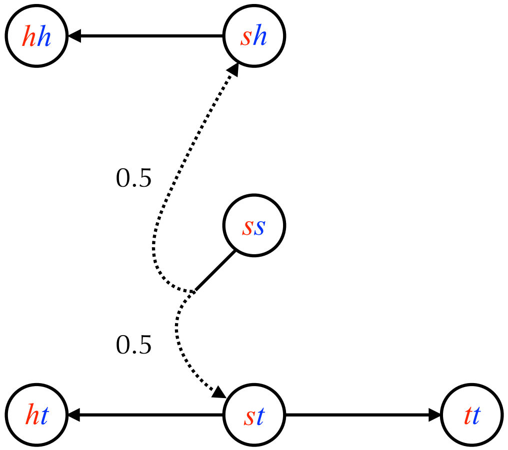
Guess




Coin




MDP from composition




$$\mathfrak{S}(s)(a) = \sum_{i=1}^{n} \mathfrak{S}_c(s)(M_i) \cdot \mathfrak{S}_{M_i}(s \downarrow_{M_i})(a)$$

global scheduler

component scheduler

local scheduler

can only see the local projection

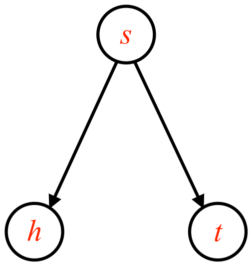
heads $= \mathfrak{S}_G(sh \downarrow_G) = \mathfrak{S}_G(s) = \mathfrak{S}_G(st \downarrow_G)$

CONICET

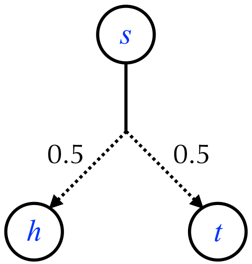
UNC

# Distributed schedulers

Guess




Coin




$$\mathfrak{S}(s)(a) = \sum_{i=1}^{n} \mathfrak{S}_c(s)(M_i) \cdot \mathfrak{S}_{M_i}(s \downarrow_{M_i})(a)$$

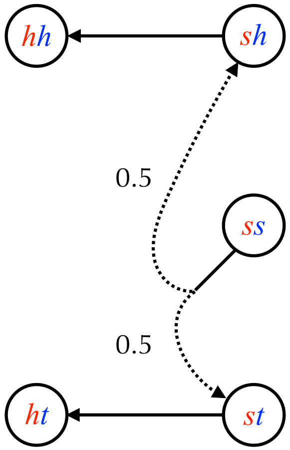
global scheduler

component scheduler

local scheduler

can only see the local projection

MDP from composition




max prob of guessing = 0.5

heads $= \mathfrak{S}_G(sh \downarrow_G) = \mathfrak{S}_G(s) = \mathfrak{S}_G(st \downarrow_G)$

CONICET

UNC

## First technique revisited

# Local decisions using RUCoP (L-RUCoP)

**Input:** number of copies $N$, target node $T$

**Output:** A routing table $LTr_n$ for each node $n$

**for all** $c \leq N$ **do**
  $(S_c, Tr_c, Pr_c) \leftarrow RUCoP(G, c, T)$
**end for**
**for all** node $n$, time slot $ts$, and $c \leq N$ **do**
  $s \leftarrow Safe\_state(n, c, ts)$
  **if** $s \in S_c$ **then**
    $LTr_n(ts, c, ts) \leftarrow \{(k, r) \in Tr_c(s) \mid first(r) = n\}$
    $ts' \leftarrow ts$
    $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, c, ts'))?\ k : 0$
    **while** $rc > 0$ **do**
      $s' \leftarrow Post(LTr_n(ts, rc, ts'))$
      $ts' = ts' + 1$
      **if** $s' \in S_{rc}$ **then**
        $LTr_n(ts, rc, ts') \leftarrow \{(k, r) \in Tr_{rc}(s') \mid first(r) = n\}$
      **else**
        **break**
      **end if**
      $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, rc, ts'))?\ k : 0$
    **end while**
  **end if**
**end for**
**return** $LTr_n$, for all node $n$.

## First technique revisited

# Local decisions using RUCoP (L-RUCoP)

**Input:** number of copies $N$, target node $T$

**Output:** A routing table $LTr_n$ for each node $n$

Construct all RUCoP tables for $c \leq N$

**for all** $c \leq N$ **do**
  $(S_c, Tr_c, Pr_c) \leftarrow RUCoP(G, c, T)$
**end for**
**for all** node $n$, time slot $ts$, and $c \leq N$ **do**
  $s \leftarrow Safe\_state(n, c, ts)$
  **if** $s \in S_c$ **then**
    $LTr_n(ts, c, ts) \leftarrow \{(k, r) \in Tr_c(s) \mid first(r) = n\}$
    $ts' \leftarrow ts$
    $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, c, ts'))?\ k : 0$
    **while** $rc > 0$ **do**
      $s' \leftarrow Post(LTr_n(ts, rc, ts'))$
      $ts' = ts' + 1$
      **if** $s' \in S_{rc}$ **then**
        $LTr_n(ts, rc, ts') \leftarrow \{(k, r) \in Tr_{rc}(s') \mid first(r) = n\}$
      **else**
        **break**
      **end if**
      $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, rc, ts'))?\ k : 0$
    **end while**
  **end if**
**end for**
**return** $LTr_n$, for all node $n$.

# First technique revisited

# Local decisions using RUCoP (L-RUCoP)

**Input:** number of copies $N$, target node $T$
**Output:** A routing table $LTr_n$ for each node $n$

**for all** $c \leq N$ **do**
    $(S_c, Tr_c, Pr_c) \leftarrow RUCoP(G, c, T)$
**end for**
**for all** node $n$, time slot $ts$, and $c \leq N$ **do**
    $s \leftarrow Safe\_state(n, c, ts)$
    **if** $s \in S_c$ **then**
        $LTr_n(ts, c, ts) \leftarrow \{(k, r) \in Tr_c(s) \mid first(r) = n\}$
        $ts' \leftarrow ts$
        $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, c, ts'))?\ k : 0$
        **while** $rc > 0$ **do**
            $s' \leftarrow Post(LTr_n(ts, rc, ts'))$
            $ts' = ts' + 1$
            **if** $s' \in S_{rc}$ **then**
                $LTr_n(ts, rc, ts') \leftarrow \{(k, r) \in Tr_{rc}(s') \mid first(r) = n\}$
            **else**
                **break**
            **end if**
            $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, rc, ts'))?\ k : 0$
        **end while**
    **end if**
**end for**
**return** $LTr_n$, for all node $n$.

Start from a safe state for node $n$ with $c$ copies at time slot $ts$

$$Safe\_state(A, 2, t_0) = [A^2 B^0 C^0 D^0 \mid t_0]$$

$$Safe\_state(A, 1, t_2) = [A^1 B^0 C^0 D^0 \mid t_2]$$

UNC

## First technique revisited

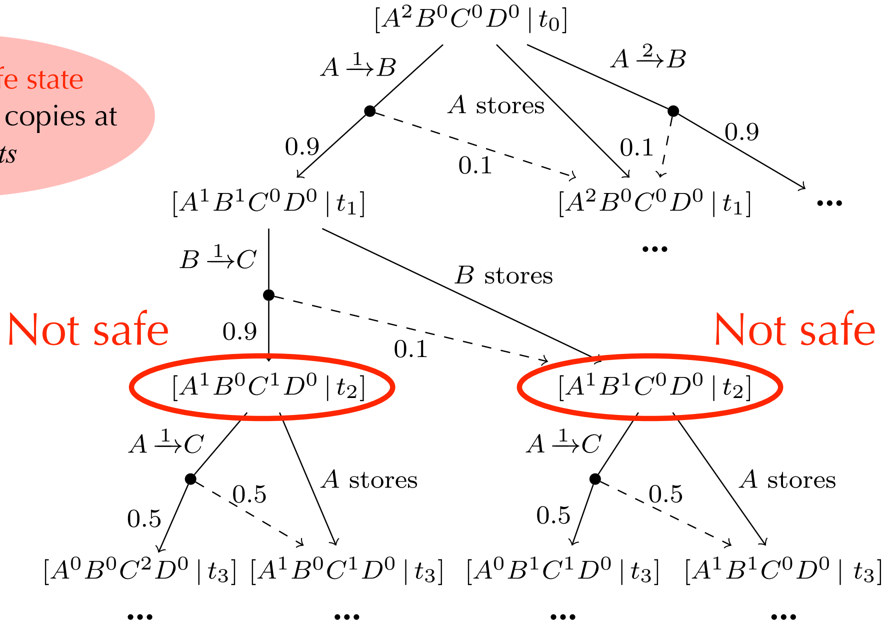
# Local decisions using RUCoP (L-RUCoP)

**Input:** number of copies $N$, target node $T$

**Output:** A routing table $LTr_n$ for each node $n$

**for all** $c \leq N$ **do**
    $(S_c, Tr_c, Pr_c) \leftarrow RUCoP(G, c, T)$
**end for**
**for all** node $n$, time slot $ts$, and $c \leq N$ **do**
    $s \leftarrow Safe\_state(n, c, ts)$
    **if** $s \in S_c$ **then**
        $LTr_n(ts, c, ts) \leftarrow \{(k, r) \in Tr_c(s) \mid first(r) = n\}$
        $ts' \leftarrow ts$
        $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, c, ts'))?\ k : 0$
        **while** $rc > 0$ **do**
            $s' \leftarrow Post(LTr_n(ts, rc, ts'))$
            $ts' = ts' + 1$
            **if** $s' \in S_{rc}$ **then**
                $LTr_n(ts, rc, ts') \leftarrow \{(k, r) \in Tr_{rc}(s') \mid first(r) = n\}$
            **else**
                **break**
            **end if**
            $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, rc, ts'))?\ k : 0$
        **end while**
    **end if**
**end for**
**return** $LTr_n$, for all node $n$.

Start from a safe state for node $n$ with $c$ copies at time slot $ts$



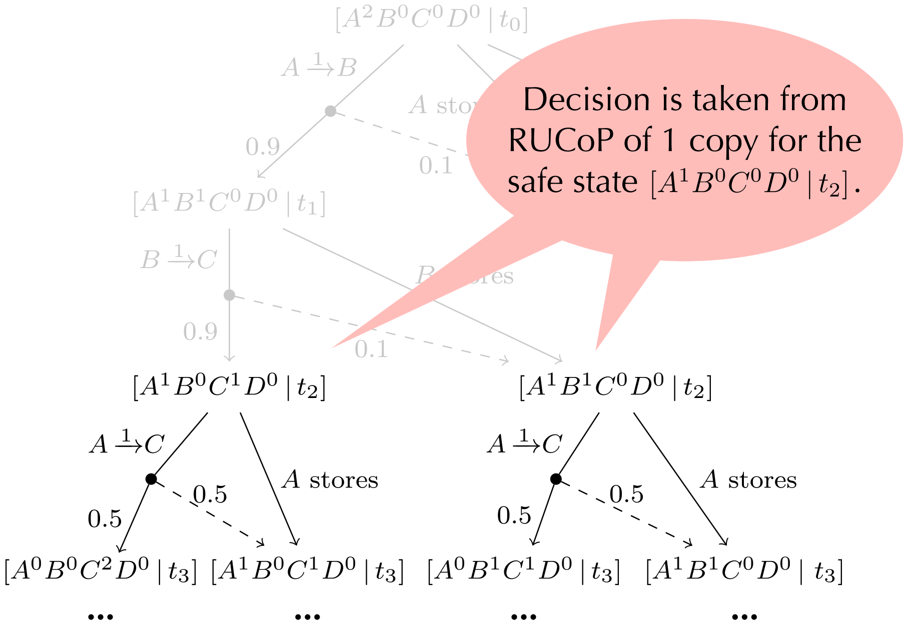
## First technique revisited

# Local decisions using RUCoP (L-RUCoP)

**Input:** number of copies $N$, target node $T$

**Output:** A routing table $LTr_n$ for each node $n$

**for all** $c \leq N$ **do**
  $(S_c, Tr_c, Pr_c) \leftarrow RUCoP(G, c, T)$
**end for**
**for all** node $n$, time slot $ts$, and $c \leq N$ **do**
  $s \leftarrow Safe\_state(n, c, ts)$
  **if** $s \in S_c$ **then**
    $LTr_n(ts, c, ts) \leftarrow \{(k, r) \in Tr_c(s) \mid first(r) = n\}$
    $ts' \leftarrow ts$
    $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, c, ts'))?\ k : 0$
    **while** $rc > 0$ **do**
      $s' \leftarrow Post(LTr_n(ts, rc, ts'))$
      $ts' = ts' + 1$
      **if** $s' \in S_{rc}$ **then**
        $LTr_n(ts, rc, ts') \leftarrow \{(k, r) \in Tr_{rc}(s') \mid first(r) = n\}$
      **else**
        **break**
      **end if**
      $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, rc, ts'))?\ k : 0$
    **end while**
  **end if**
**end for**
**return** $LTr_n$, for all node $n$.



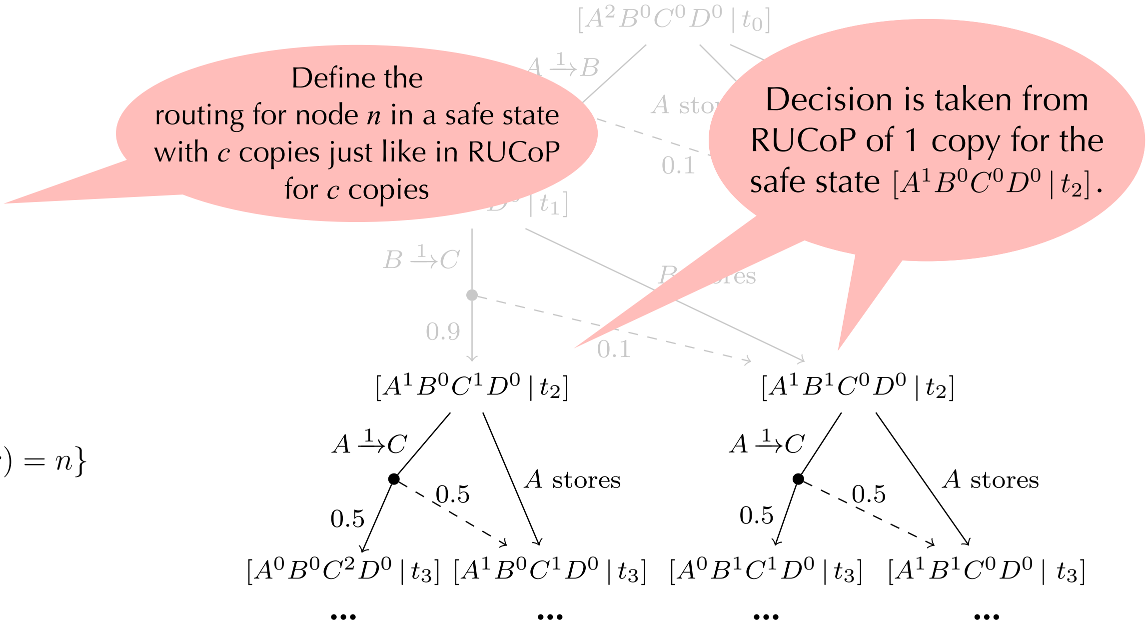
## First technique revisited

# Local decisions using RUCoP (L-RUCoP)

**Input:** number of copies $N$, target node $T$

**Output:** A routing table $LTr_n$ for each node $n$

**for all** $c \leq N$ **do**
  $(S_c, Tr_c, Pr_c) \leftarrow RUCoP(G, c, T)$
**end for**
**for all** node $n$, time slot $ts$, and $c \leq N$ **do**
  $s \leftarrow Safe\_state(n, c, ts)$
  **if** $s \in S_c$ **then**
    $LTr_n(ts, c, ts) \leftarrow \{(k, r) \in Tr_c(s) \mid first(r) = n\}$
    $ts' \leftarrow ts$
    $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, c, ts'))?\ k : 0$
    **while** $rc > 0$ **do**
      $s' \leftarrow Post(LTr_n(ts, rc, ts'))$
      $ts' = ts' + 1$
      **if** $s' \in S_{rc}$ **then**
        $LTr_n(ts, rc, ts') \leftarrow \{(k, r) \in Tr_{rc}(s') \mid first(r) = n\}$
      **else**
        **break**
      **end if**
      $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, rc, ts'))?\ k : 0$
    **end while**
  **end if**
**end for**
**return** $LTr_n$, for all node $n$.



First technique revisited

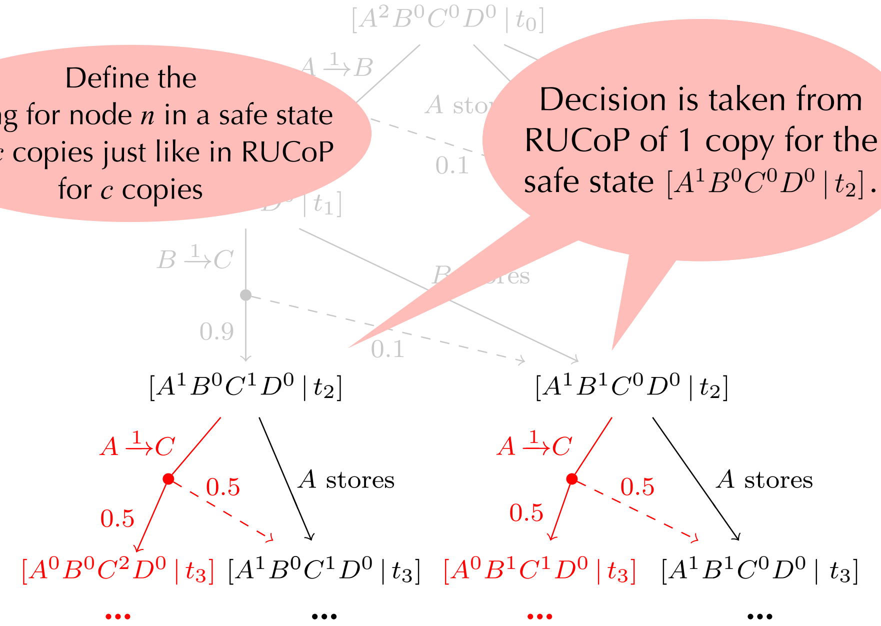
# Local decisions using RUCoP (L-RUCoP)

**Input:** number of copies $N$, target node $T$
**Output:** A routing table $LTr_n$ for each node $n$

**for all** $c \leq N$ **do**
    $(S_c, Tr_c, Pr_c) \leftarrow RUCoP(G, c, T)$
**end for**
**for all** node $n$, time slot $ts$, and $c \leq N$ **do**
    $s \leftarrow Safe\_state(n, c, ts)$
    **if** $s \in S_c$ **then**
        $LTr_n(ts, c, ts) \leftarrow \{(k, r) \in Tr_c(s) \mid first(r) = n\}$
        $ts' \leftarrow ts$
        $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, c, ts'))?\ k : 0$
        **while** $rc > 0$ **do**
            $s' \leftarrow Post(LTr_n(ts, rc, ts'))$
            $ts' = ts' + 1$
            **if** $s' \in S_{rc}$ **then**
                $LTr_n(ts, rc, ts') \leftarrow \{(k, r) \in Tr_{rc}(s') \mid first(r) = n\}$
            **else**
                **break**
            **end if**
            $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, rc, ts'))?\ k : 0$
        **end while**
    **end if**
**end for**
**return** $LTr_n$, for all node $n$.

Define the routing for node $n$ in a safe state with $c$ copies just like in RUCoP for $c$ copies

Decision is taken from RUCoP of 1 copy for the safe state $[A^1 B^0 C^0 D^0 \mid t_2]$.

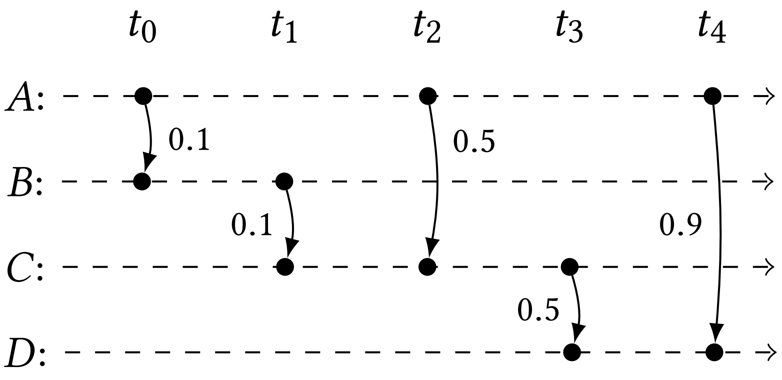


## First technique revisited

# Local decisions using RUCoP (L-RUCoP)

**Input:** number of copies $N$, target node $T$
**Output:** A routing table $LTr_n$ for each node $n$

**for all** $c \leq N$ **do**
  $(S_c, Tr_c, Pr_c) \leftarrow RUCoP(G, c, T)$
**end for**
**for all** node $n$, time slot $ts$, and $c \leq N$ **do**
  $s \leftarrow Safe\_state(n, c, ts)$
  **if** $s \in S_c$ **then**
    $LTr_n(ts, c, ts) \leftarrow \{(k, r) \in Tr_c(s) \mid first(r) = n\}$
    $ts' \leftarrow ts$
    $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, c, ts'))?\ k : 0$
    **while** $rc > 0$ **do**
      $s' \leftarrow Post(LTr_n(ts, rc, ts'))$
      $ts' = ts' + 1$
      **if** $s' \in S_{rc}$ **then**
        $LTr_n(ts, rc, ts') \leftarrow \{(k, r) \in Tr_{rc}(s') \mid first(r) = n\}$
      **else**
        **break**
      **end if**
      $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, rc, ts'))?\ k : 0$
    **end while**
  **end if**
**end for**
**return** $LTr_n$, for all node $n$.



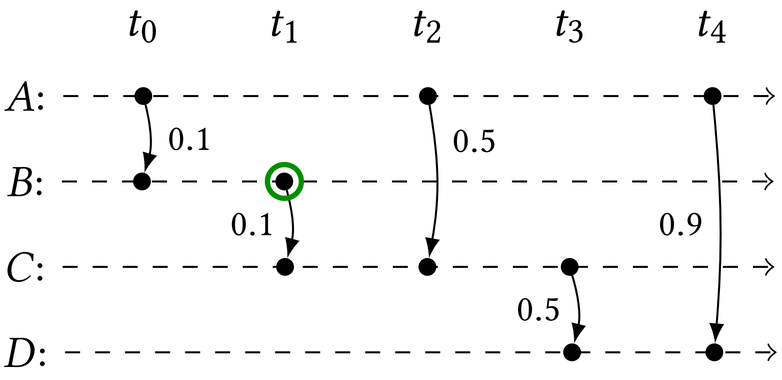

Sometimes a node has some information about other nodes (e.g. when it just sent a message)

First technique revisited

# Local decisions using RUCoP (L-RUCoP)

**Input:** number of copies $N$, target node $T$
**Output:** A routing table $LTr_n$ for each node $n$

**for all** $c \leq N$ **do**
  $(S_c, Tr_c, Pr_c) \leftarrow RUCoP(G, c, T)$
**end for**
**for all** node $n$, time slot $ts$, and $c \leq N$ **do**
  $s \leftarrow Safe\_state(n, c, ts)$
  **if** $s \in S_c$ **then**
    $LTr_n(ts, c, ts) \leftarrow \{(k, r) \in Tr_c(s) \mid first(r) = n\}$
    $ts' \leftarrow ts$
    $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, c, ts'))?\ k : 0$
    **while** $rc > 0$ **do**
      $s' \leftarrow Post(LTr_n(ts, rc, ts'))$
      $ts' = ts' + 1$
      **if** $s' \in S_{rc}$ **then**
        $LTr_n(ts, rc, ts') \leftarrow \{(k, r) \in Tr_{rc}(s') \mid first(r) = n\}$
      **else**
        **break**
      **end if**
      $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, rc, ts'))?\ k : 0$
    **end while**
  **end if**
**end for**
**return** $LTr_n$, for all node $n$.




$t_1$: $B$ sends a copy to $C$ who ack reception

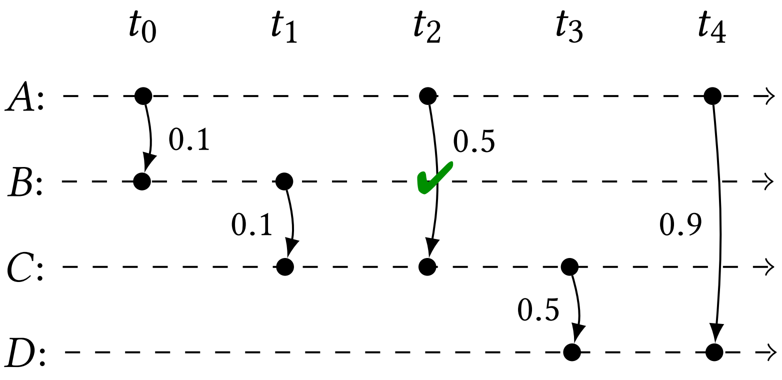
Sometimes a node has some information about other nodes (e.g. when it just sent a message)

# First technique revisited

# Local decisions using RUCoP (L-RUCoP)

**Input:** number of copies $N$, target node $T$
**Output:** A routing table $LTr_n$ for each node $n$

**for all** $c \leq N$ **do**
  $(S_c, Tr_c, Pr_c) \leftarrow RUCoP(G, c, T)$
**end for**
**for all** node $n$, time slot $ts$, and $c \leq N$ **do**
  $s \leftarrow Safe\_state(n, c, ts)$
  **if** $s \in S_c$ **then**
    $LTr_n(ts, c, ts) \leftarrow \{(k, r) \in Tr_c(s) \mid first(r) = n\}$
    $ts' \leftarrow ts$
    $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, c, ts'))?\ k : 0$
    **while** $rc > 0$ **do**
      $s' \leftarrow Post(LTr_n(ts, rc, ts'))$
      $ts' = ts' + 1$
      **if** $s' \in S_{rc}$ **then**
        $LTr_n(ts, rc, ts') \leftarrow \{(k, r) \in Tr_{rc}(s') \mid first(r) = n\}$
      **else**
        **break**
      **end if**
      $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, rc, ts'))?\ k : 0$
    **end while**
  **end if**
**end for**
**return** $LTr_n$, for all node $n$.




$t_2$: $B$ knows $C$ has a copy

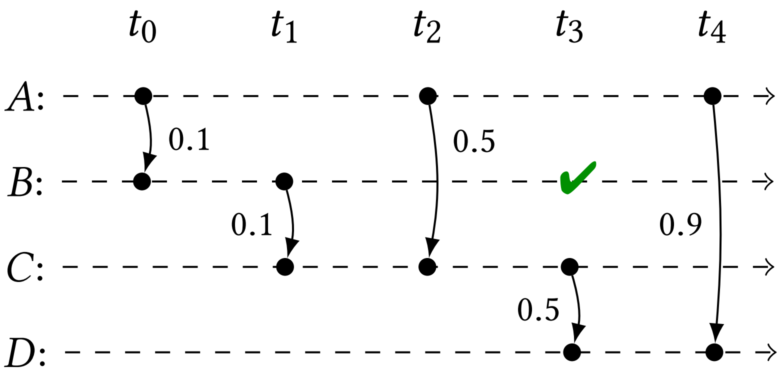
Sometimes a node has some information about other nodes (e.g. when it just sent a message)

First technique revisited

# Local decisions using RUCoP (L-RUCoP)

**Input:** number of copies $N$, target node $T$
**Output:** A routing table $LTr_n$ for each node $n$

**for all** $c \leq N$ **do**
  $(S_c, Tr_c, Pr_c) \leftarrow RUCoP(G, c, T)$
**end for**
**for all** node $n$, time slot $ts$, and $c \leq N$ **do**
  $s \leftarrow Safe\_state(n, c, ts)$
  **if** $s \in S_c$ **then**
    $LTr_n(ts, c, ts) \leftarrow \{(k, r) \in Tr_c(s) \mid first(r) = n\}$
    $ts' \leftarrow ts$
    $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, c, ts'))?\ k : 0$
    **while** $rc > 0$ **do**
      $s' \leftarrow Post(LTr_n(ts, rc, ts'))$
      $ts' = ts' + 1$
      **if** $s' \in S_{rc}$ **then**
        $LTr_n(ts, rc, ts') \leftarrow \{(k, r) \in Tr_{rc}(s') \mid first(r) = n\}$
      **else**
        **break**
      **end if**
      $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, rc, ts'))?\ k : 0$
    **end while**
  **end if**
**end for**
**return** $LTr_n$, for all node $n$.




$t_3$: $B$ knows $C$ has a copy

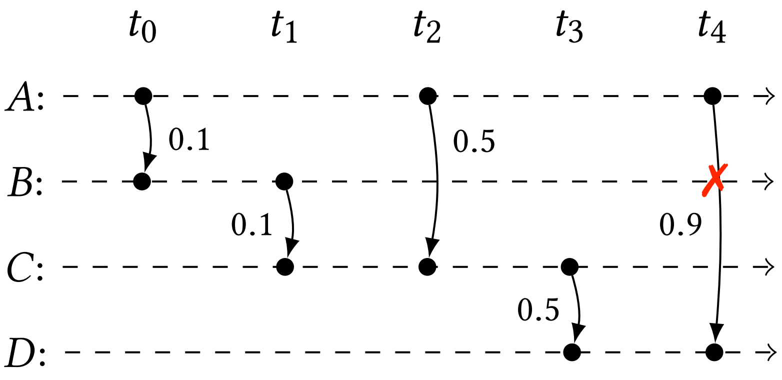
Sometimes a node has some information about other nodes (e.g. when it just sent a message)

## First technique revisited

# Local decisions using RUCoP (L-RUCoP)

**Input:** number of copies $N$, target node $T$
**Output:** A routing table $LTr_n$ for each node $n$

**for all** $c \leq N$ **do**
    $(S_c, Tr_c, Pr_c) \leftarrow RUCoP(G, c, T)$
**end for**
**for all** node $n$, time slot $ts$, and $c \leq N$ **do**
    $s \leftarrow Safe\_state(n, c, ts)$
    **if** $s \in S_c$ **then**
        $LTr_n(ts, c, ts) \leftarrow \{(k, r) \in Tr_c(s) \mid first(r) = n\}$
        $ts' \leftarrow ts$
        $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, c, ts'))?\ k : 0$
        **while** $rc > 0$ **do**
            $s' \leftarrow Post(LTr_n(ts, rc, ts'))$
            $ts' = ts' + 1$
            **if** $s' \in S_{rc}$ **then**
                $LTr_n(ts, rc, ts') \leftarrow \{(k, r) \in Tr_{rc}(s') \mid first(r) = n\}$
            **else**
                **break**
            **end if**
            $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, rc, ts'))?\ k : 0$
        **end while**
    **end if**
**end for**
**return** $LTr_n$, for all node $n$.




$t_4$: $B$ does not know if $C$ has a copy

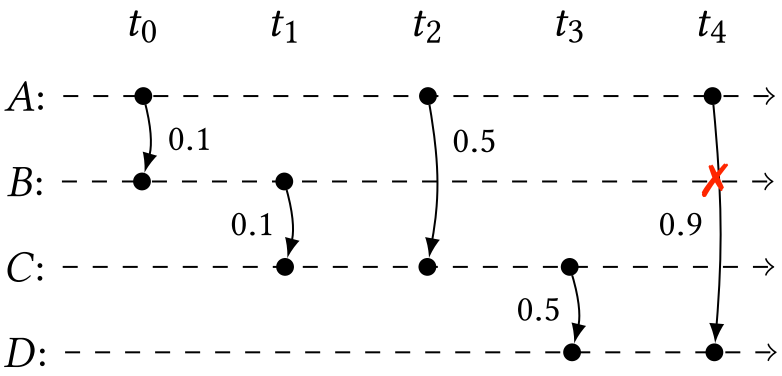
Sometimes a node has some information about other nodes (e.g. when it just sent a message)

## First technique revisited

# Local decisions using RUCoP (L-RUCoP)

**Input:** number of copies $N$, target node $T$
**Output:** A routing table $LTr_n$ for each node $n$

**for all** $c \leq N$ **do**
  $(S_c, Tr_c, Pr_c) \leftarrow RUCoP(G, c, T)$
**end for**
**for all** node $n$, time slot $ts$, and $c \leq N$ **do**
  $s \leftarrow Safe\_state(n, c, ts)$
  **if** $s \in S_c$ **then**
    $LTr_n(ts, c, ts) \leftarrow \{(k, r) \in Tr_c(s) \mid first(r) = n\}$
    $ts' \leftarrow ts$
    $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, c, ts'))?\ k : 0$
    **while** $rc > 0$ **do**
      $s' \leftarrow Post(LTr_n(ts, rc, ts'))$
      $ts' = ts' + 1$
      **if** $s' \in S_{rc}$ **then**
        $LTr_n(ts, rc, ts') \leftarrow \{(k, r) \in Tr_{rc}(s') \mid first(r) = n\}$
      **else**
        **break**
      **end if**
      $rc \leftarrow (\exists\ (k, n) \in LTr_n(ts, rc, ts'))?\ k : 0$
    **end while**
  **end if**
**end for**
**return** $LTr_n$, for all node $n$.




$t_4$: $B$ does not know if $C$ has a copy

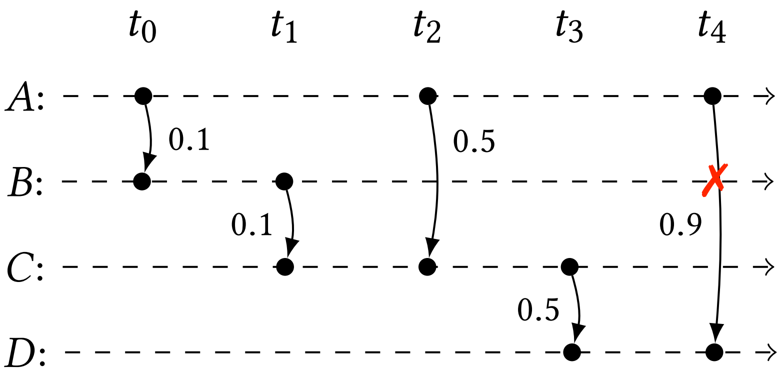
# First technique revisited

# Local decisions using RUCoP (L-RUCoP)

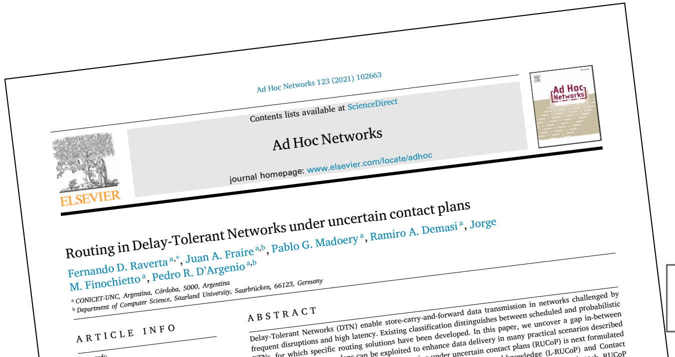
**Input:** number of copies $N$, target node $T$

**Output:** A routing table $LTr_n$ for each node $n$

```
for all c ≤ N do
    (S_c, Tr_c, Pr_c) ← RUCoP(G, c, T)
end for
for all node n, time slot ts, and c ≤ N do
    s ← Safe_state(n, c, ts)
    if s ∈ S_c then
        LTr_n(ts, c, ts) ← {(k, r) ∈ Tr_c(s) | first(r) = n}
        ts' ← ts
        rc ← (∃ (k, n) ∈ LTr_n(ts, c, ts'))? k : 0
        while rc > 0 do
            s' ← Post(LTr_n(ts, rc, ts'))
            ts' = ts' + 1
            if s' ∈ S_rc then
                LTr_n(ts, rc, ts') ← {(k, r) ∈ Tr_rc(s') | first(r) = n}
            else
                break
            end if
            rc ← (∃ (k, n) ∈ LTr_n(ts, rc, ts'))? k : 0
        end while
    end if
end for
return LTr_n, for all node n.
```




$t_4$: $B$ does not know if $C$ has a copy


Ad Hoc Networks 123 (2021) 102663

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

ELSEVIER

Routing in Delay-Tolerant Networks under uncertain contact plans

Fernando D. Raverta [a,*], Juan A. Fraire [a,b], Pablo G. Madoery [a], Ramiro A. Demasi [a], Jorge M. Finochietto [a], Pedro R. D'Argenio [a,b]

[a] CONICET-UNC, Argentina, Córdoba, 5000, Argentina

[b] Department of Computer Science, Saarland University, Saarbrücken, 66123, Germany

ARTICLE INFO

ABSTRACT

Delay-Tolerant Networks (DTN) enable store-carry-and-forward data transmission in networks challenged by frequent disruptions and high latency. Existing classification distinguishes between scheduled and probabilistic DTNs, for which specific routing solutions have been developed. In this paper, we uncover a gap in-between ...

Second technique revisited

# SMC + LSS of distributed schedulers

- Resolving non-determinism in SMC+LSS

$$\mathcal{H}(\sigma.s) \bmod n$$

32-bit hash function

state as a bit vector

number of choices at $s$

## Second technique revisited

# SMC + LSS of distributed schedulers

- Resolving non-determinism in SMC+LSS

$$\mathcal{H}(\sigma.s) \bmod n$$

- Resolving non-determinism in SMC+LSS+DS

$$\mathcal{H}(\sigma.(s{\downarrow_{M_i}})) \bmod n_i$$

bit vector limited to component $i$

number of choices of component $i$ at $s$

**Input:** Network of VMDP $M = \|_{SV}(M_1, \ldots, M_n)$ with $[\![M]\!] = \langle S, s_I, A, T \rangle$, goal set $G \subseteq S$, $\sigma \in \mathbb{Z}_{32}$, $\mathcal{H}$ uniform deterministic, PRNG $\mathcal{U}_{\text{pr}}$.

$s := s_I$

**while** $s \notin G$ **do** *// break on goal state*

  **if** $\forall\, s \xrightarrow{a} \mu \colon \mu = \{\, s \mapsto 1 \,\}$ **then break** *// break on self-loops*

  $C := \{\, j \mid T(s) \cap I_t(M_j) \neq \emptyset \,\}$ *// get active components*

  $i := \mathcal{U}_{\text{pr}}(\{\, j \mapsto \frac{1}{|C|} \mid j \in C \,\})$ *// select component uniformly*

  $T_i := T(s) \cap I_t(M_i)$ *// get component's transitions*

  $\langle a, \mu \rangle := (\mathcal{H}(\sigma.s{\downarrow_{M_i}}) \bmod |T_i|)$-th element of $T_i$ *// schedule local transition*

  $s := \mathcal{U}_{\text{pr}}(\mu)$ *// select next state according to $\mu$*

**return** $s \in G$

CONICET

UNC

## Second technique revisited

# SMC + LSS of distributed schedulers

- Resolving non-determinism in SMC+LSS

$$\mathcal{H}(\sigma.s) \bmod n$$

- Resolving non-determinism in SMC+LSS+DS

$$\mathcal{H}(\sigma.(s{\downarrow}_{M_i})) \bmod n_i$$

bit vector limited to component *i*

number of choices of component *i* at *s*

**Input:** Network of VMDP $M = \|_{SV}(M_1, \ldots, M_n)$ with $[\![M]\!] = \langle S, s_I, A, T\rangle$,
goal set $G \subseteq S$, $\sigma \in \mathbb{Z}_{32}$, $\mathcal{H}$ uniform deterministic, PRNG $\mathcal{U}_{\text{pr}}$.

$s := s_I$

**while** $s \notin G$ **do** — *// break on goal state*

- **if** $\forall\, s \xrightarrow{a} \mu\colon \mu = \{\, s \mapsto 1 \,\}$ **then break** — *// break on self-loops*
- $C := \{\, j \mid T(s) \cap I_t(M_j) \neq \emptyset \,\}$ — *// get active components*
- $i := \mathcal{U}_{\text{pr}}(\{\, j \mapsto \frac{1}{|C|} \mid j \in C \,\})$ — *// select component uniformly*
- $T_i := T(s) \cap I_t(M_i)$ — *// get component's transitions*
- $\langle a, \mu\rangle := (\mathcal{H}(\sigma.s{\downarrow}_{M_i}) \bmod |T_i|)$-th element of $T_i$ — *// schedule local transition*
- $s := \mathcal{U}_{\text{pr}}(\mu)$ — *// select next state according to $\mu$*

**return** $s \in G$

CONICET

UNC

Second technique revisited

# SMC + LSS of distributed schedulers

- Resolving non-determinism in SMC+LSS

$$\mathcal{H}(\sigma.s) \bmod n$$

- Resolving non-determinism in SMC+LSS+DS

$$\mathcal{H}(\sigma.(s{\downarrow}_{M_i})) \bmod n_i$$

**Input:** Network of VMDP $M = \|_{SV}(M_1, \ldots, M_n)$ with $[\![M]\!] = \langle S, s_I, A, T \rangle$,
goal set $G \subseteq S$, $\sigma \in \mathbb{Z}_{32}$, $\mathcal{H}$ uniform deterministic, PRNG $\mathcal{U}_{\text{pr}}$.

$s := s_I$

**while** $s \notin G$ **do** // *break on goal state*

  **if** $\forall\, s \xrightarrow{a} \mu\colon \mu = \{\, s \mapsto 1 \,\}$ **then break** // *break on self-loops*

  $C := \{\, j \mid T(s) \cap I_t(M_j) \neq \emptyset \,\}$ // *get active components*

  $i := \mathcal{U}_{\text{pr}}(\{\, j \mapsto \frac{1}{|C|} \mid j \in C \,\})$ // *select component uniformly*

  $T_i := T(s) \cap I_t(M_i)$ // *get component's transitions*

  $\langle a, \mu \rangle := (\mathcal{H}(\sigma.s{\downarrow}_{M_i}) \bmod |T_i|)$-th element of $T_i$ // *schedule local transition*

  $s := \mathcal{U}_{\text{pr}}(\mu)$ // *select next state according to* $\mu$

**return** $s \in G$

A network of MDP $M = M_1 \,||\, \ldots \,||\, M_n$ is good for distributed scheduling w.r.t. reachability of goal set $G$ if in all states $s \in S$ of $[\![M]\!] = \langle S, s_I, A, T \rangle$ where $|T(s)| > 1 \wedge |\{\, i \mid T(s) \cap I_t(M_i) \neq \emptyset \,\}| > 1$ we have

- $\forall\, s \xrightarrow{a} s'\colon s \in G \Leftrightarrow s' \in G$,
- $\forall\, i \in \{\, 1, \ldots, n \,\}\colon\ |I_t(M_i) \cap T(s)| > 1 \ \Rightarrow\ I_c(I_t(M_i) \cap T(s)) = \{\, M_i \,\}$, and
- $s \xrightarrow{a} s' \ \Rightarrow\ \forall\, M_c \in \{\, M_1, \ldots, M_n \,\} \setminus I_c(s \xrightarrow{a} s')\colon s{\downarrow}_{M_c} = s'{\downarrow}_{M_c}$.

stuttering

no component disables actions in another component

idle components do not change their state

CONICET

UNC

Second technique revisited

# SMC + LSS of distributed schedulers

- Resolving non-determinism in SMC+LSS

$$\mathcal{H}(\sigma.s) \bmod n$$

- Resolving non-determinism in SMC+LSS+DS

$$\mathcal{H}(\sigma.(s{\downarrow_{M_i}})) \bmod n_i$$

**Input:** Network of VMDP $M = \|_{SV}(M_1, \ldots, M_n)$ with $[\![M]\!] = \langle S, s_I, A, T\rangle$, goal set $G \subseteq S$, $\sigma \in \mathbb{Z}_{32}$, $\mathcal{H}$ uniform deterministic, PRNG $\mathcal{U}_{\mathrm{pr}}$.

$s := s_I$
**while** $s \notin G$ **do** // *break on goal state*
  **if** $\forall s \xrightarrow{a} \mu\colon \mu = \{ s \mapsto 1 \}$ **then break** // *break on self-loops*
  $C := \{ j \mid T(s) \cap I_t(M_j) \neq \emptyset \}$ // *get active components*
  $i := \mathcal{U}_{\mathrm{pr}}(\{ j \mapsto \frac{1}{|C|} \mid j \in C \})$ // *select component uniformly*
  $T_i := T(s) \cap I_t(M_i)$ // *get component's transitions*
  $\langle a, \mu\rangle := (\mathcal{H}(\sigma.s{\downarrow_{M_i}}) \bmod |T_i|)$-th element of $T_i$ // *schedule local transition*
  $s := \mathcal{U}_{\mathrm{pr}}(\mu)$ // *select next state according to* $\mu$
**return** $s \in G$

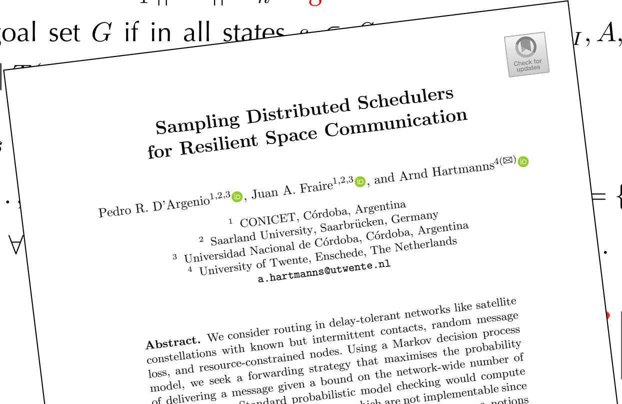
A network of MDP $M = M_1 \| \ldots \| M_n$ is good for distributed scheduling w.r.t. reachability of goal set $G$ if in all states $s$ … $_I, A, T\rangle$ where $|T(s)| > 1 \wedge |\{ i$ …

- $\forall s \xrightarrow{a} s'\colon s$ …
- $\forall i \in \{1, \ldots$ … $= \{ M_i \}$, and
- $s \xrightarrow{a} s' \Rightarrow \forall$ … .

stuttering

no component disables actions in another component

idle components do not change their state


Sampling Distributed Schedulers for Resilient Space Communication

Pedro R. D'Argenio[1,2,3], Juan A. Fraire[1,2,3], and Arnd Hartmanns[4(✉)]

[1] CONICET, Córdoba, Argentina
[2] Saarland University, Saarbrücken, Germany
[3] Universidad Nacional de Córdoba, Córdoba, Argentina
[4] University of Twente, Enschede, The Netherlands
a.hartmanns@utwente.nl

**Abstract.** We consider routing in delay-tolerant networks like satellite constellations with known but intermittent contacts, random message loss, and resource-constrained nodes. Using a Markov decision process model, we seek a forwarding strategy that maximises the probability of delivering a message given a bound on the network-wide number of … Standard probabilistic model checking would compute … which are not implementable since … notions

CONICET

UNC

Third technique revisited

# Reinforcement Learning with Q-Learning

One $Q_{M_i} : S_{M_i} \times Act \to [0,1]$ for each component $M_i$

**for** $j := 1$ **to** $nr\_episodes$ **do**
  Episode$(s_I, \epsilon_j, \alpha_j)$

Episode$(s,\epsilon,\alpha)$
  $\langle a, \mu \rangle :=$ sample uniformly from $Act(s)$
    $\oplus_\epsilon \operatorname*{arg\,max}_{\langle a',\mu' \rangle \in Act(s)} Q(s, \langle a', \mu' \rangle)$    *// choose with probability* $\epsilon$
  $s' := \mathcal{U}_{\text{pr}}(\mu)$    *// select next state according to* $\mu$
  **if** $\forall\, s \xrightarrow{a} \mu \colon \mu = \{ s \mapsto 1 \}$ **then return**    *// run ended unsuccessfully*
  **else if** $s \in G$ **then return**    *// run reached the goal*
  **else** Episode$(s', \epsilon, \alpha)$
  **forall** component $M_i$ **do**
    $Q_{M_i}(s\downarrow_{M_i}, \langle a, \mu \rangle) := (1-\alpha) \cdot Q_{M_i}(s, \langle a, \mu \rangle)$    *// update matrix* $Q_{M_i}$
      $+\ \alpha \cdot \left( \mathbf{1}_G(s'\downarrow_{M_i}) + \max_{\langle a',\mu' \rangle \in Act(s'\downarrow_{M_i})} Q_{M_i}(s'\downarrow_{M_i}, \langle a', \mu' \rangle) \right)$

CONICET

UNC

Third technique revisited

# Reinforcement Learning with Q-Learning

One $Q_{M_i} : S_{M_i} \times Act \to [0,1]$ for each component $M_i$

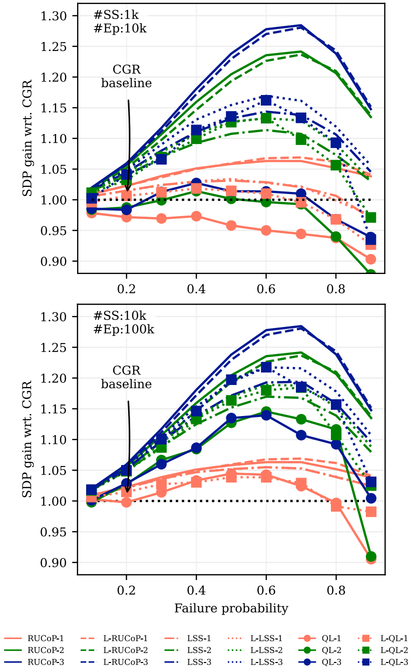
**for** $j := 1$ **to** $nr\_episodes$ **do**
  Episode$(s_I, \epsilon_j, \alpha_j)$

Episode$(s,\epsilon,\alpha)$
  $\langle a, \mu\rangle :=$ sample uniformly from $Act(s)$
    $\oplus_\epsilon \operatorname*{arg\,max}_{\langle a',\mu'\rangle \in Act(s)} Q(s, \langle a', \mu'\rangle)$ *// choose with probability* $\epsilon$
  $s' := \mathcal{U}_{\mathrm{pr}}(\mu)$ *// select next state according to* $\mu$
  **if** $\forall\, s \xrightarrow{a} \mu \colon \mu = \{ s \mapsto 1 \}$ **then return** *// run ended unsuccessfully*
  **else if** $s \in G$ **then return** *// run reached the goal*
  **else** Episode$(s', \epsilon, \alpha)$
  **forall** component $M_i$ **do**
    $Q_{M_i}(s\downarrow_{M_i}, \langle a, \mu\rangle) := (1-\alpha)\cdot Q_{M_i}(s, \langle a, \mu\rangle)$ *// update matrix* $Q_{M_i}$
      $+\ \alpha \cdot \left(\mathbf{1}_G(s'\downarrow_{M_i}) + \max_{\langle a',\mu'\rangle \in Act(s'\downarrow_{M_i})} Q_{M_i}(s'\downarrow_{M_i}, \langle a', \mu'\rangle)\right)$

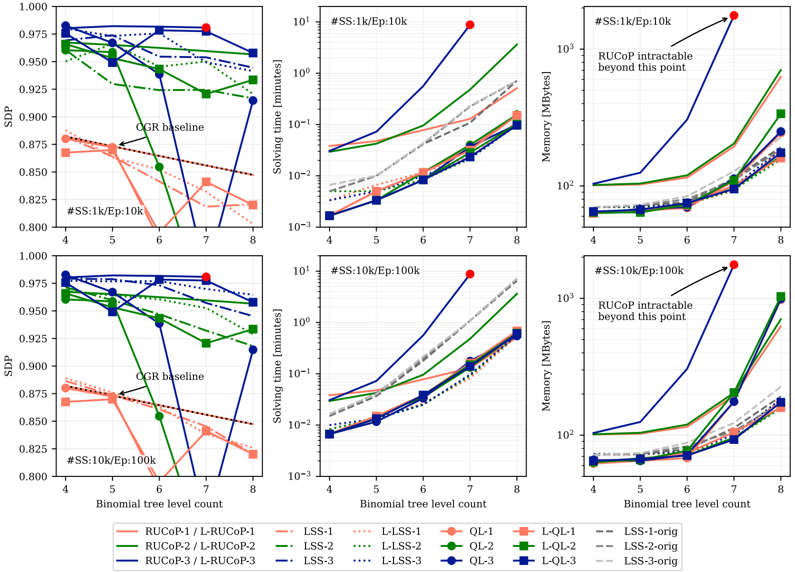
CONICET

UNC

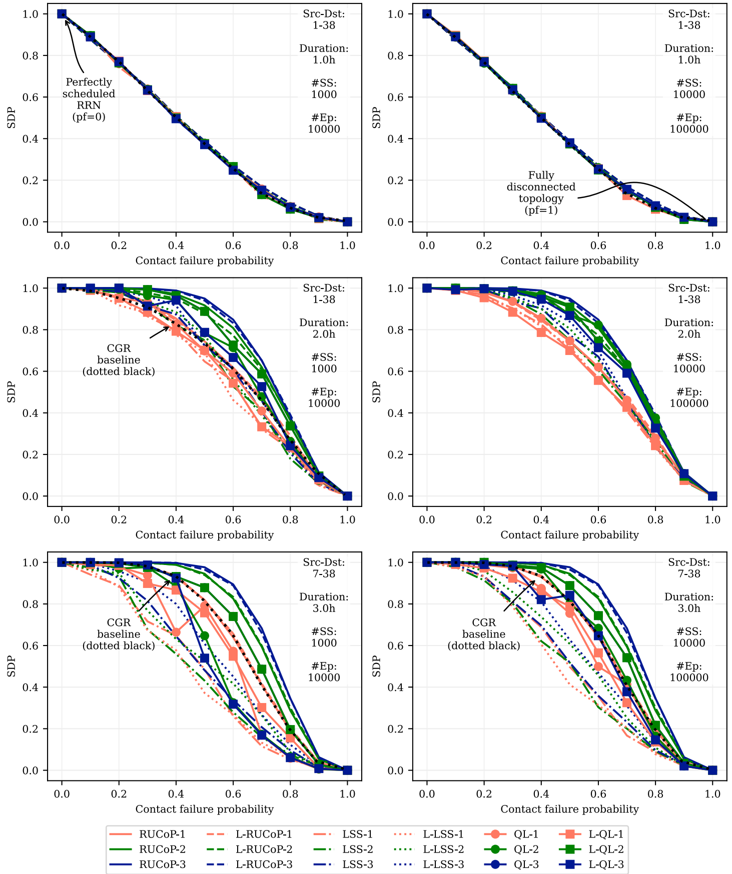
# Experiments (delivery probability)




SDP on a random network




SDP, solving time and memory on binomial networks



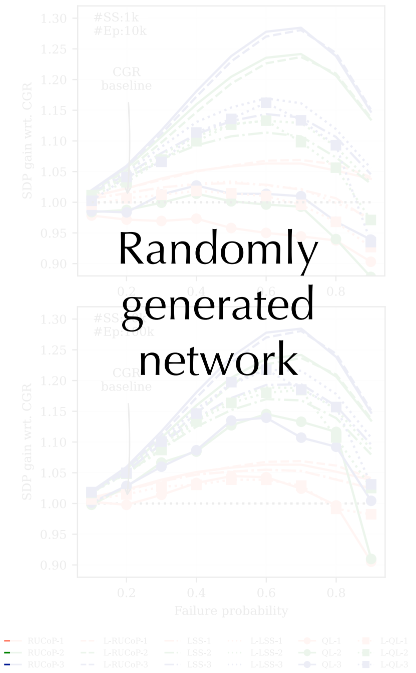

SDP for ring road networks with different contact plans

CONICET

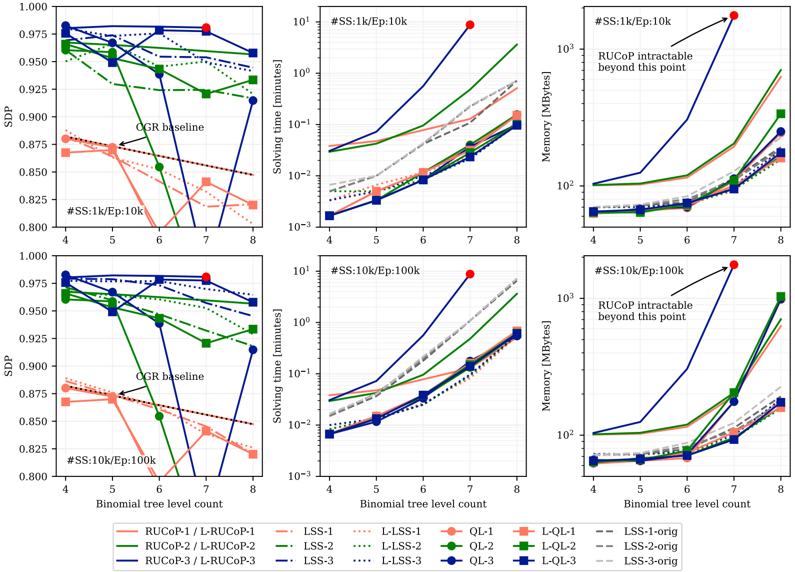
UNC

# Experiments (delivery probability)

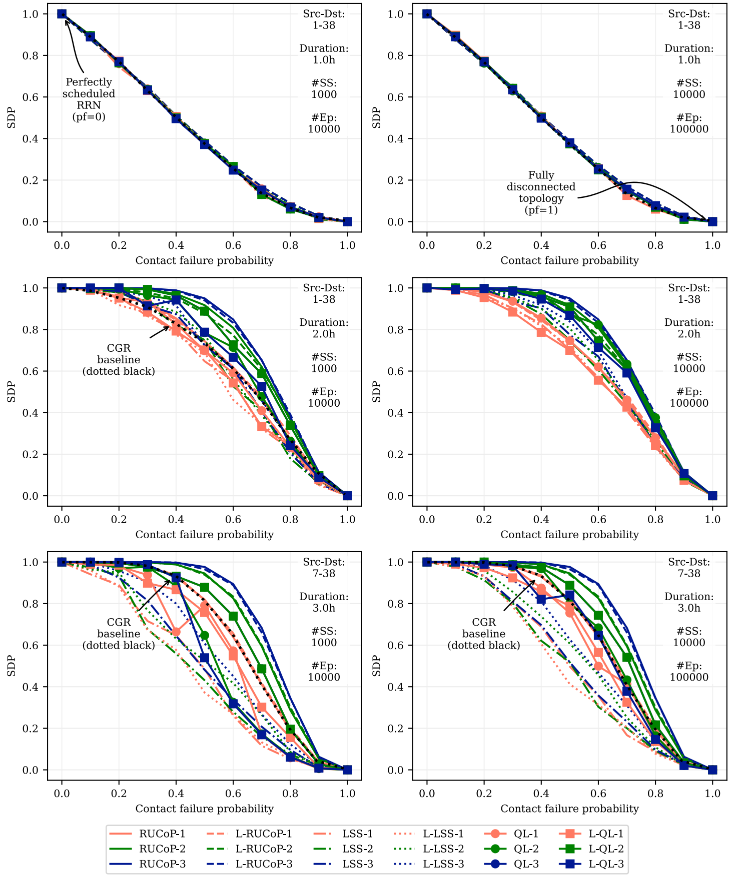
Randomly generated network



SDP on a random network




SDP, solving time and memory on binomial networks

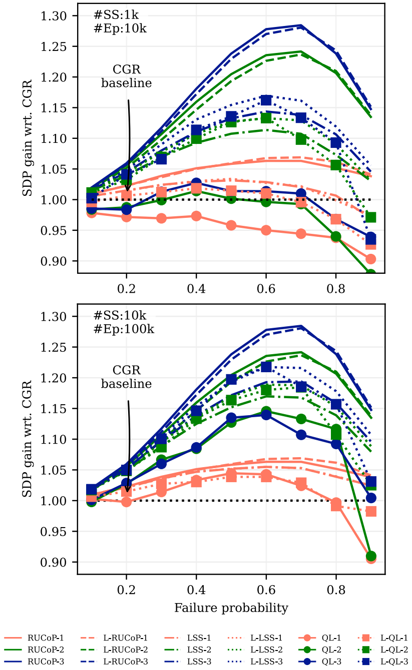



SDP for ring road networks with different contact plans

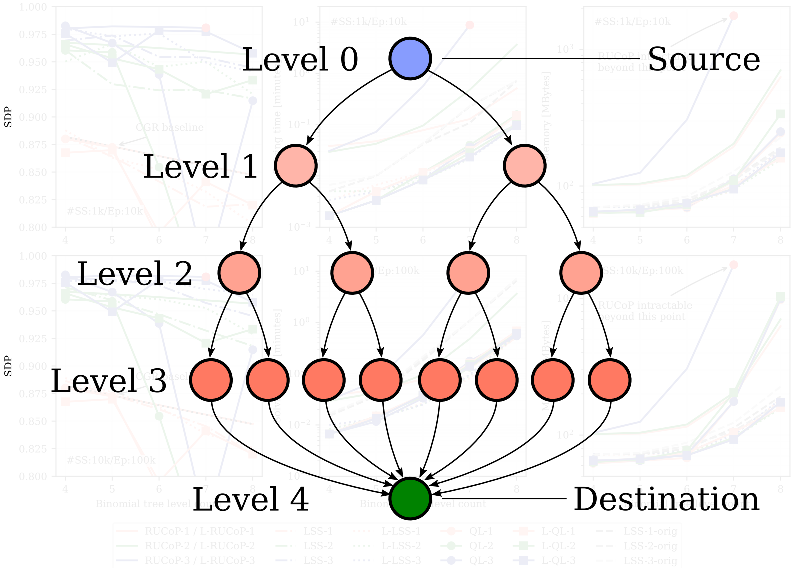
CONICET

UNC

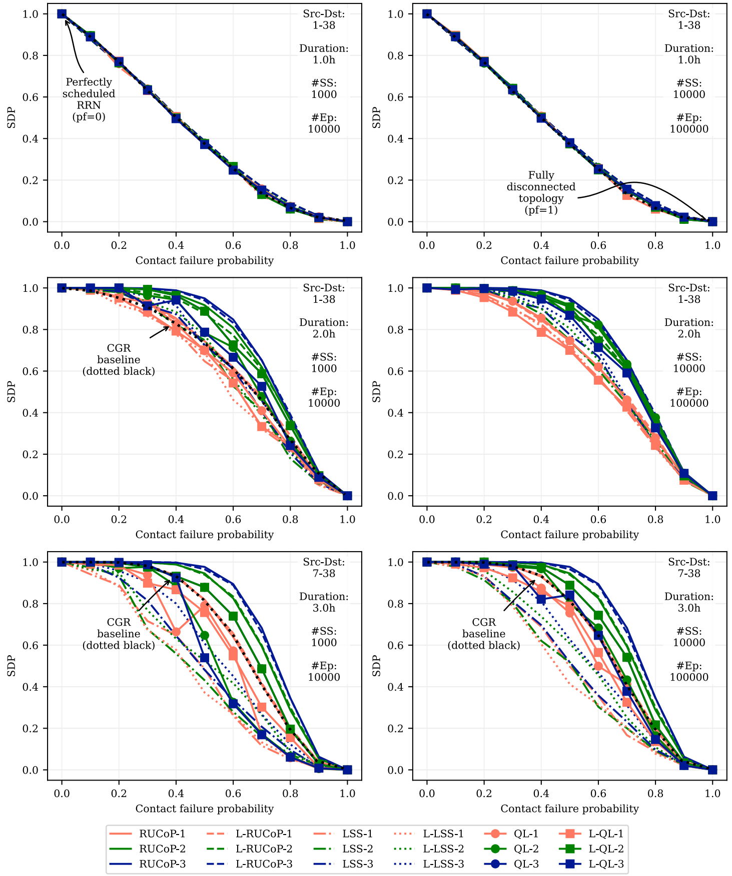
# Experiments (delivery probability)




SDP on a random network




SDP, solving time and memory on binomial networks



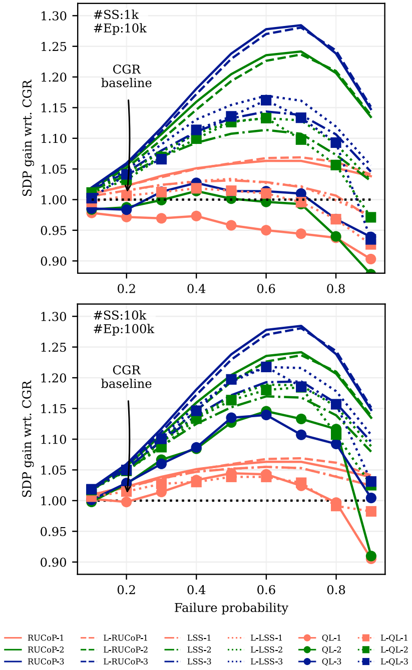

SDP for ring road networks with different contact plans

CONICET

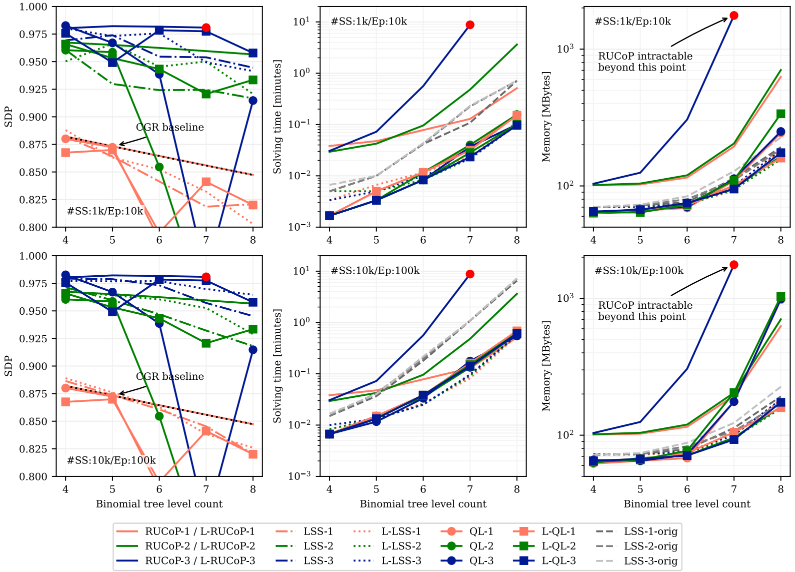
UNC

# Experiments (delivery probability)

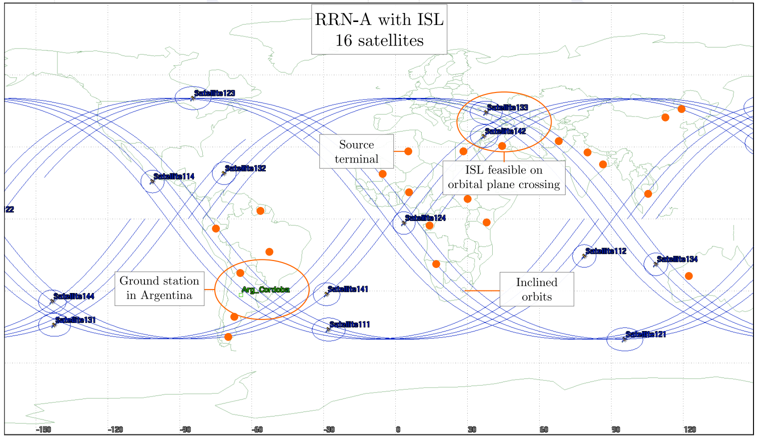
16 satellites
22 ground terminals
24 hours plan slotted in minutes
(1440 minutes)




SDP on a random network




SDP, solving time and memory on binomial networks



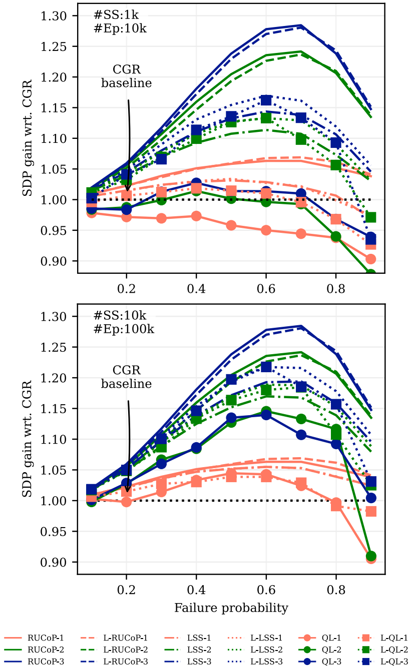

| RRN with ISL - 16 LEO satellites - 1 ground station (dst) - 22 ground terminals (src) | | | | | |
|---|---|---|---|---|---|
| **Name** | **T. Anomaly** [deg] | **Altitude** [km] | **Arg. Perigee** [deg] | **Inclination** [deg] | **RAAN** [deg] |
| Satellite111,12,13,14 | 0, 90, 180, 270 | 500 | 0 | 50 | 0 |
| Satellite121,22,23,24 | 23, 113, 203, 293 | 500 | 0 | 50 | 90 |
| Satellite131,32,33,34 | 45, 135, 225, 315 | 500 | 0 | 50 | 180 |
| Satellite141,42,43,44 | 68, 158, 248, 338 | 500 | 0 | 50 | 270 |
| **Name** | **Latitude** [deg] | **Longitude** [deg] | **Altitude** [km] | | |
| Arg_Cordoba | -31.5242 | -64.4636 | 0.724 | | |

SDP for ring road networks with different contact plans

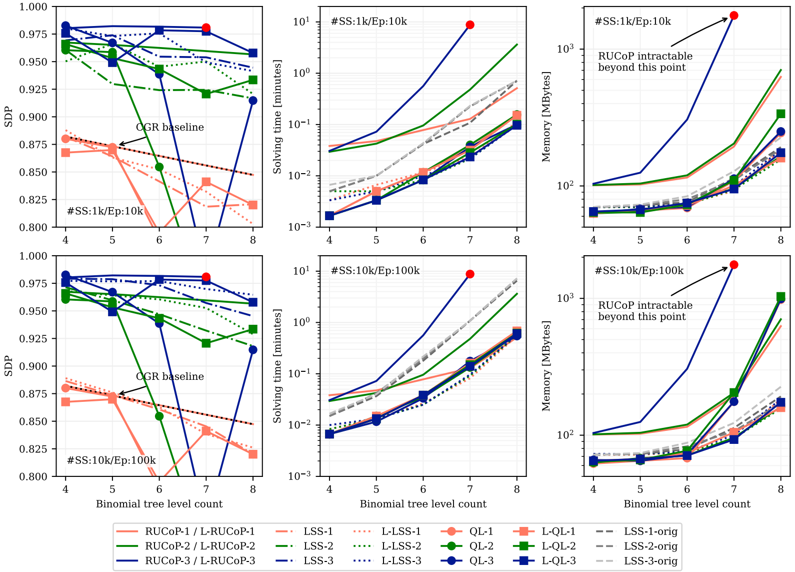
CONICET

UNC

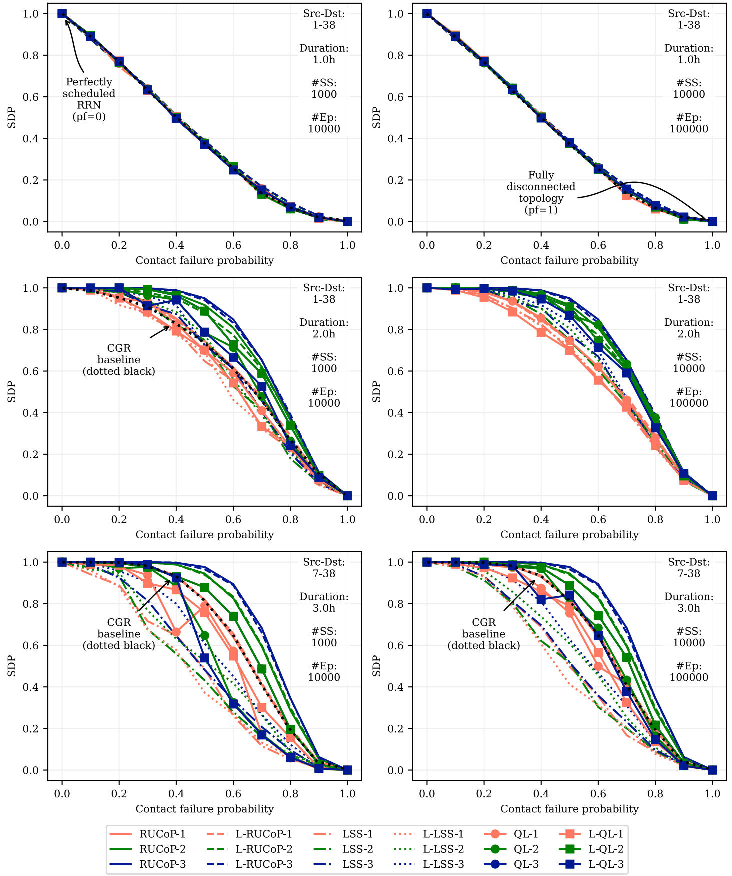
# Experiments (delivery probability)




SDP on a random network




SDP, solving time and memory on binomial networks

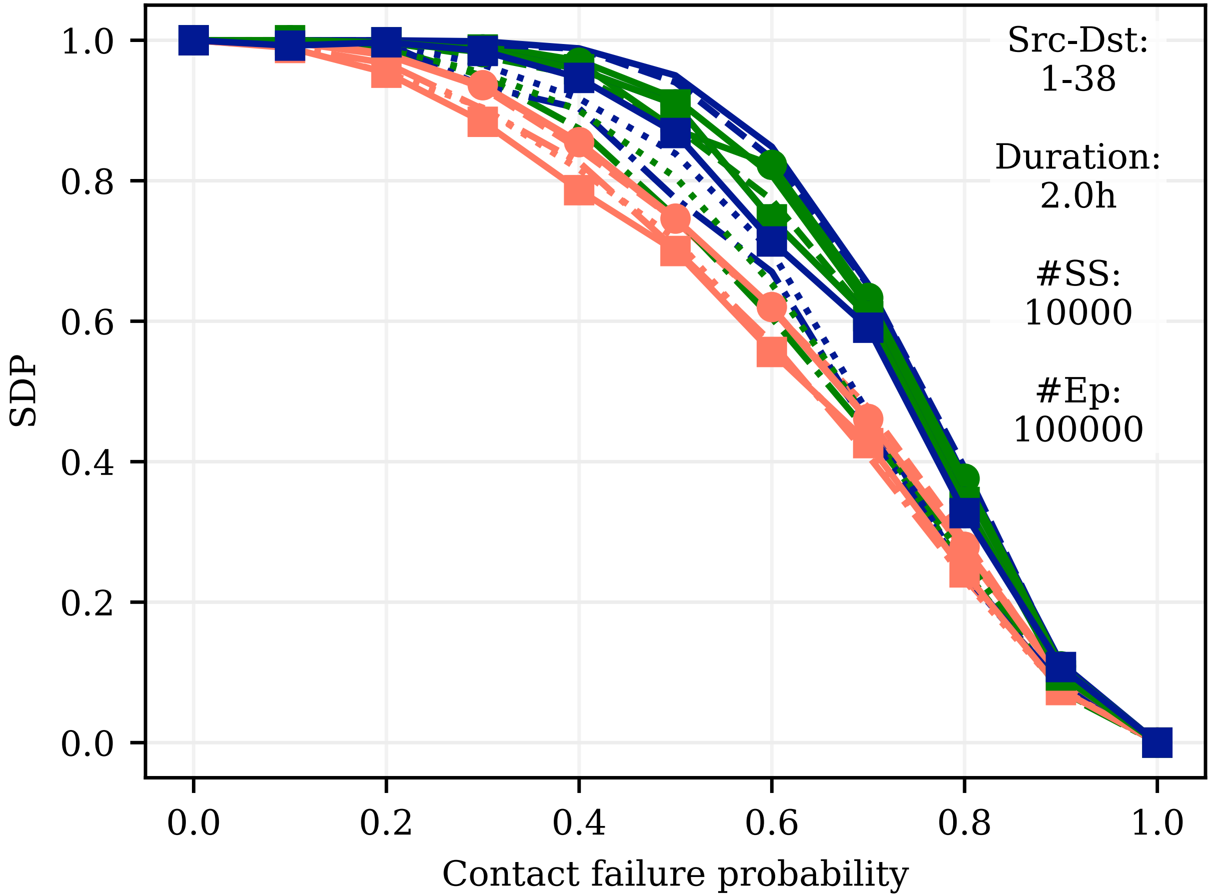



SDP for ring road networks with different contact plans

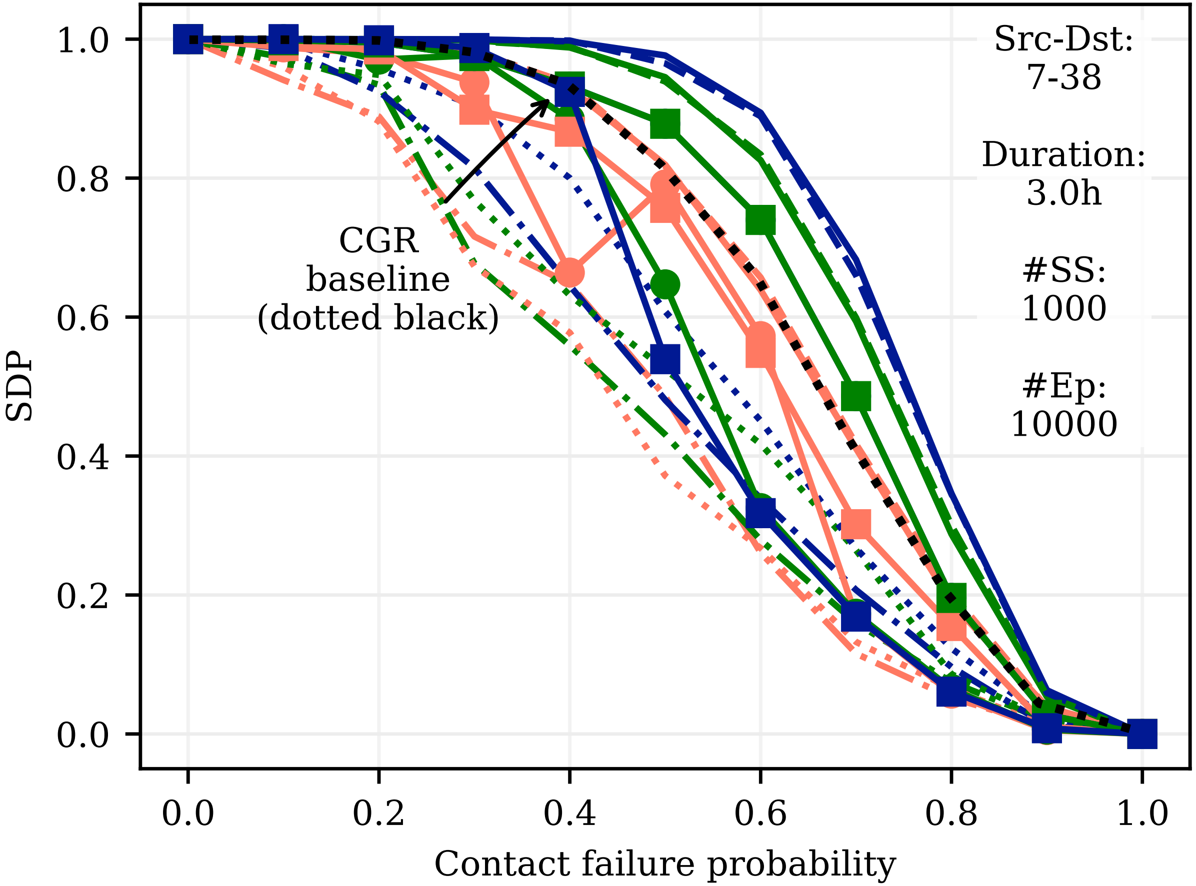
CONICET

UNC

# Experiments (delivery probability)



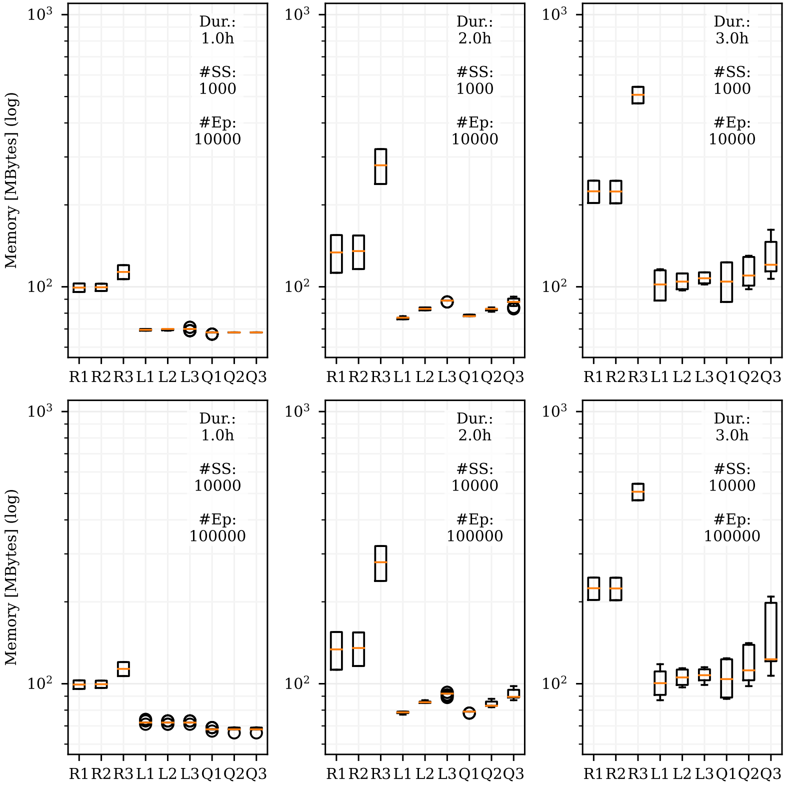
# Experiments (delivery probability)



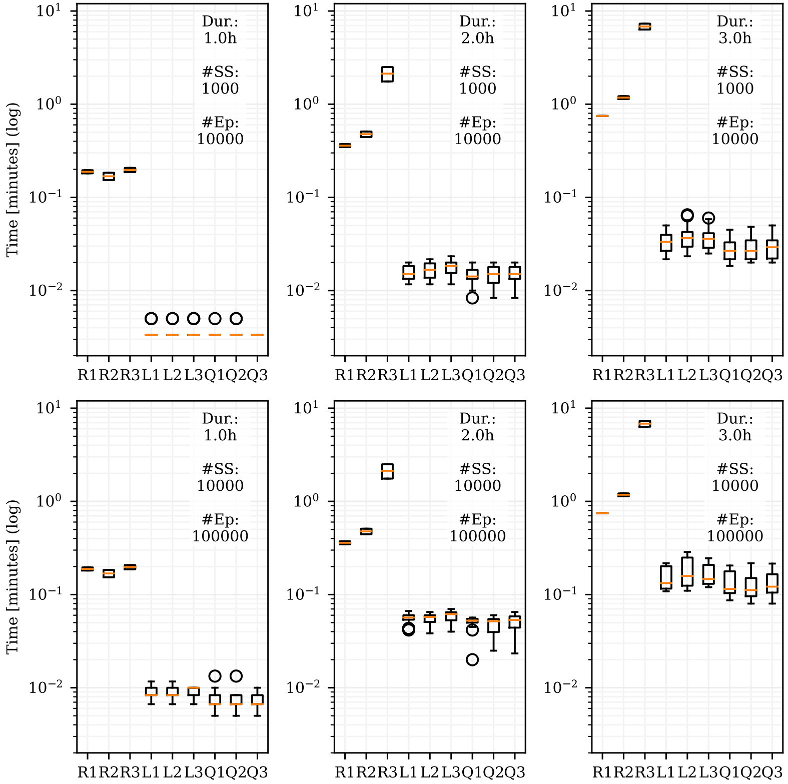

CONICET

UNC

# Experiments (RRN)




Time

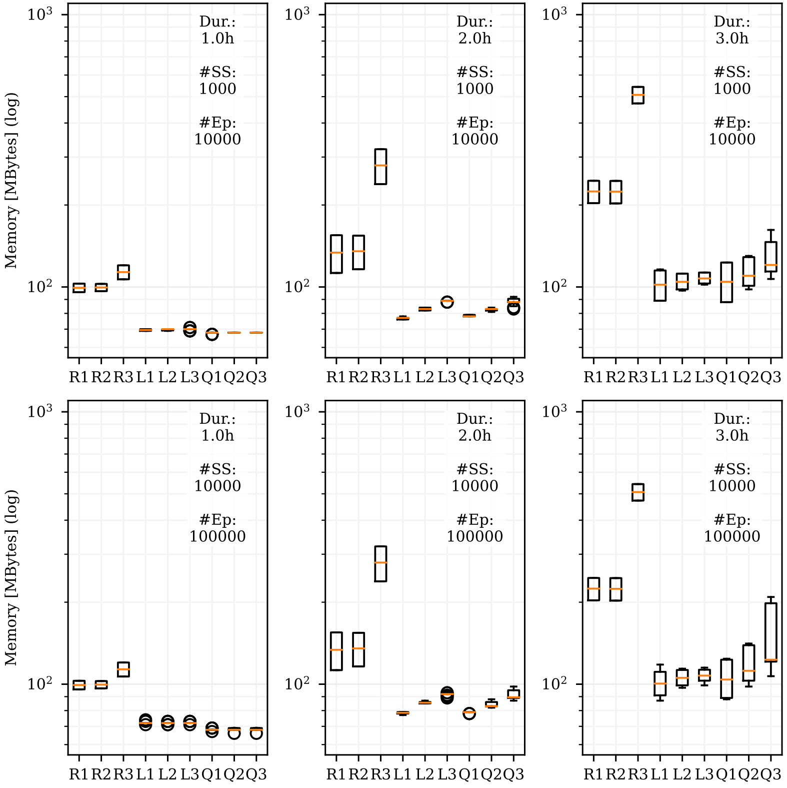



Memory

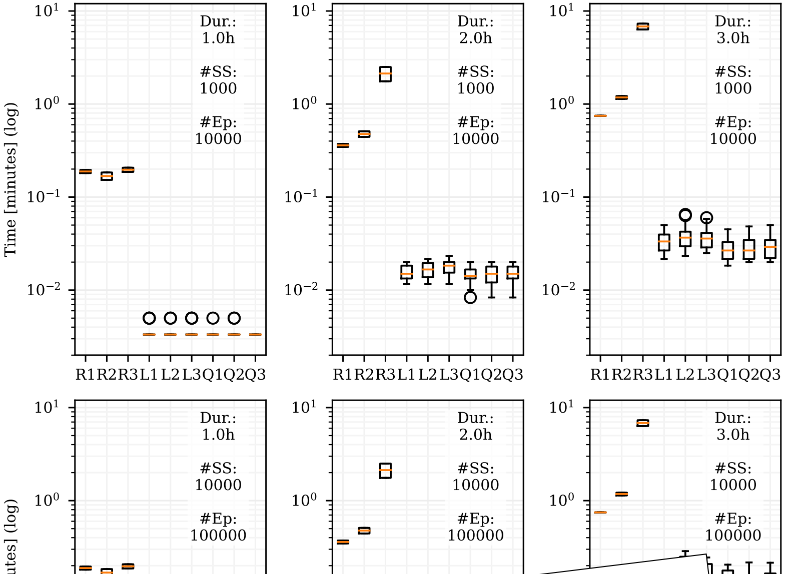
CONICET

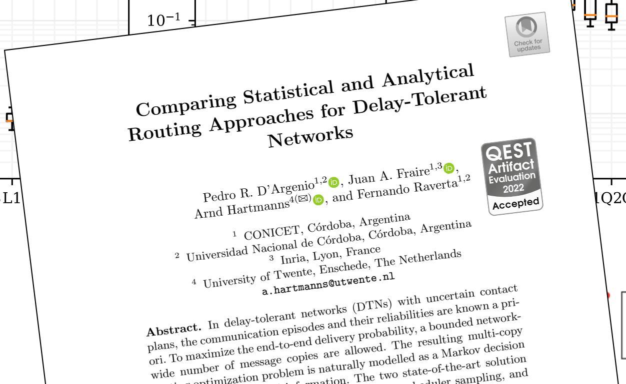
UNC

# Experiments (RRN)




Time



Comparing Statistical and Analytical Routing Approaches for Delay-Tolerant Networks

Pedro R. D'Argenio[1,2], Juan A. Fraire[1,3], Arnd Hartmanns[4](✉), and Fernando Raverta[1,2]

[1] CONICET, Córdoba, Argentina
[2] Universidad Nacional de Córdoba, Córdoba, Argentina
[3] Inria, Lyon, France
[4] University of Twente, Enschede, The Netherlands
a.hartmanns@utwente.nl

**Abstract.** In delay-tolerant networks (DTNs) with uncertain contact plans, the communication episodes and their reliabilities are known a priori. To maximize the end-to-end delivery probability, a bounded network-wide number of message copies are allowed. The resulting multi-copy routing optimization problem is naturally modelled as a Markov decision ... The two state-of-the-art solution ... scheduler sampling, and

CONICET

UNC

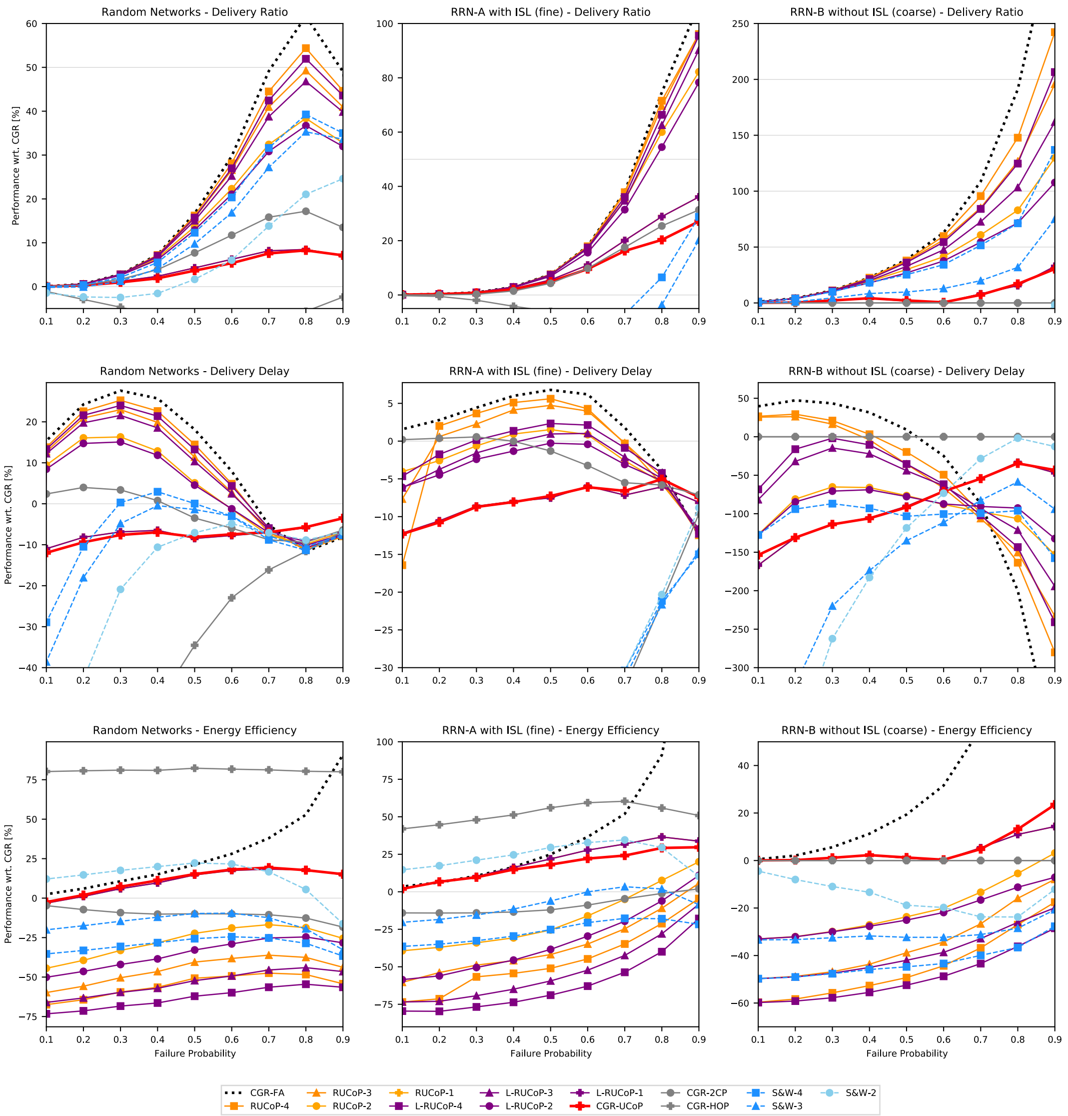
# Experiments (routing efficiency)

(Only RUCoP & L-RUCoP)

Probability

Latency

Energy




CONICET

# Concluding remarks

- Clear increase of reliability (particularly L-RUCoP & CGR-UCoP)
- Q-learning does not show consistent results 😕
- Comparison on latency is mixed. It very much depends on probability of link failure
- Particularly, (L-)RUCoP-1 & CGR-UCoP are more energy efficient than CGR
- All new algorithms are demanding:
  - Routing tables need to be calculated on ground and uploaded to the satellites
  - (CGR requires uploading the contact plan, routing decisions are made on flight)
  - CGR-UCoP requires uploading an annotated contact plan, routing decisions are made on flight. However, RUCoP is needed to annotate.

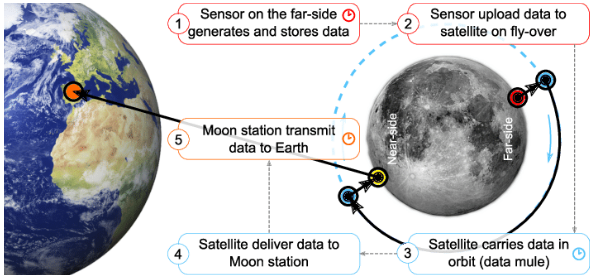
CONICET

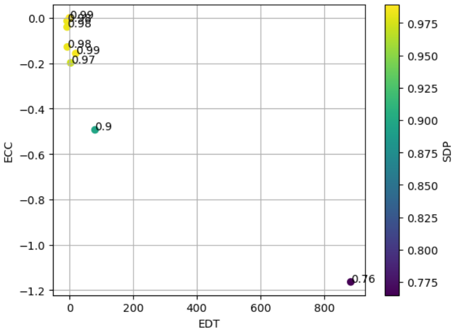
UNC

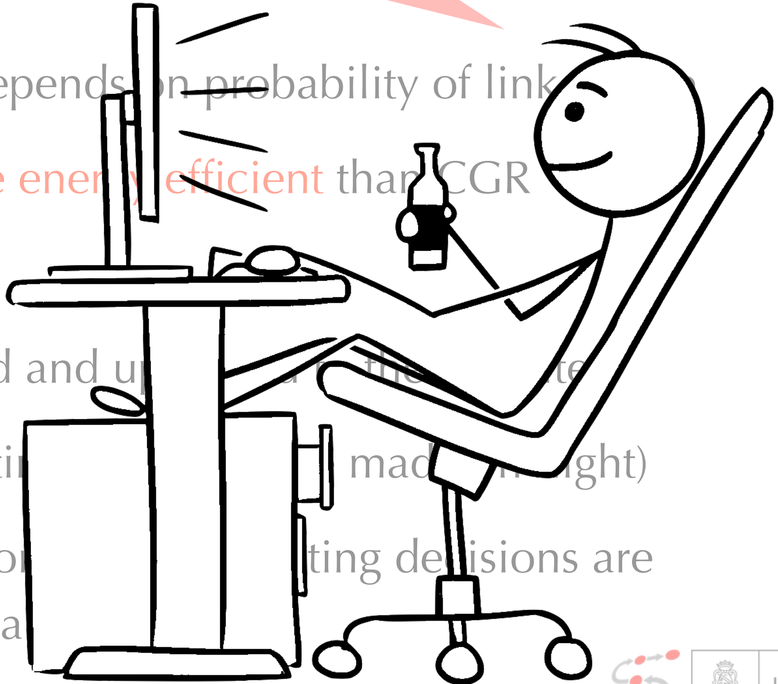
*In production:*

- Multi-objective prototyping with Storm
- Prioritized multi-objective variant of RUCoP

- Clear increase of relia bi
consistent results 🙁
mixed. It very much depends on probability of link
& CGR-UCoP are more ener efficient than CGR
nanding:
e calculated on ground and u
the contact plan, routi mad ght)
oading an annotated co ting decisions are
, RUCoP is needed to a



Picture from Feldmann, Fraire & Walter, 2018 IEEE ICC // Pareto front from Torrella 2023, MsC thesis

# Optimal Route Synthesis in Space DTN using Markov Decision Processes

Pedro R. D'Argenio

Universidad Nacional de Córdoba – CONICET
https://cs.famaf.unc.edu.ar/~dargenio/

CONICET

ICTAC 2023 - Lima

UNC