

# Synthesis of Safe, Small and Optimal Strategies for Cyber-Physical Systems

Kim G Larsen

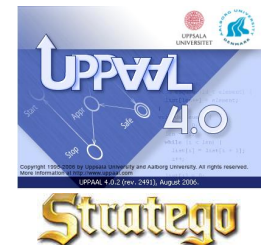
Aalborg University, DENMARK



AALBORG UNIVERSITET



VILLUM FONDEN



# Collaborators

- Alexandre David
- Marius Mikucionis
- Peter Jensen
- Jakob Taankvist
- Manfred Jaeger
- Axel Legay
- Sean Sedwards
- Pranav Ashok
- Jan Kretinsky
- Adrien Le Coënt
- Maximilian Weininger



[ATVA19] Teaching Stratego to play Ball

[QEST19] SOS: Synthesis for Hybrid MDP

[ISOLA20] Approximating Euclidain by Imprecise MDPs

[ADHS21] Learning Safe and Optimal Control m Strategies for Storm Water Detention Ponds

# Cyber Physical Systems

## Games Everywhere

### SAFE:

Find **controller** that function (correct) under stated conditions for a specified period of time.

Discrete

Real Time

### OPTIMAL:

Find a **controller** for a given system such that a certain optimality criterion is achieved.

Resources

Stochasticity

### COMPACT:

Find a representation of the **controller** that may fit a given embedded platform

Hybrid

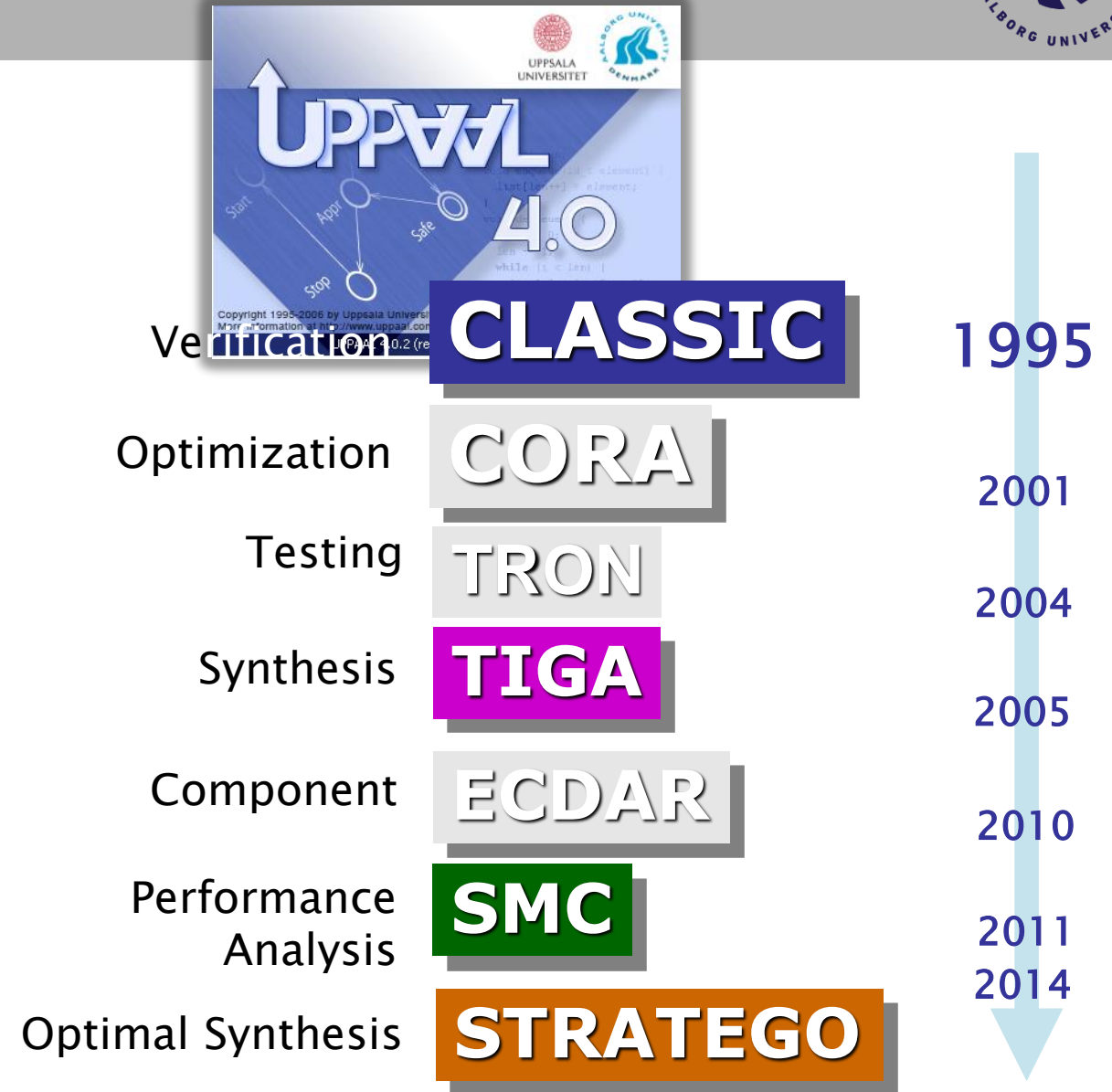
Networked ES

Cloud Computing  
Big Data

Cyber-Physical Systems

# Overview

- UPPAAL STRATEGO
  - Dagstuhl 19432  
Smart Farming Challenge
- Better Learning
- Approximation Methods
- Compact Strategies
  
- Applications
- Future Challenges



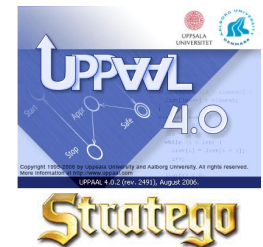
# SMART FARMING & UPPAAL STRATEGO



AALBORG UNIVERSITET



VILLUM FONDEN

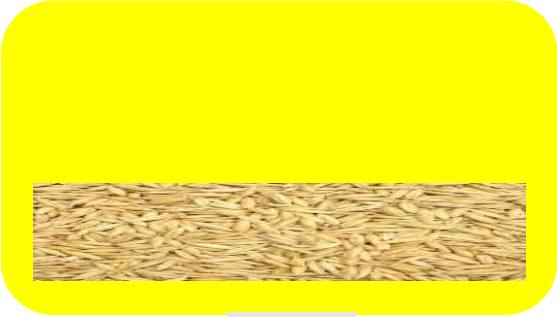


TACAS15: David, Jensen, L, Mikucionis, Taankvist: Uppaal Stratego.

# Smart Farming / Dagstuhl 19432 Oct19



Collection

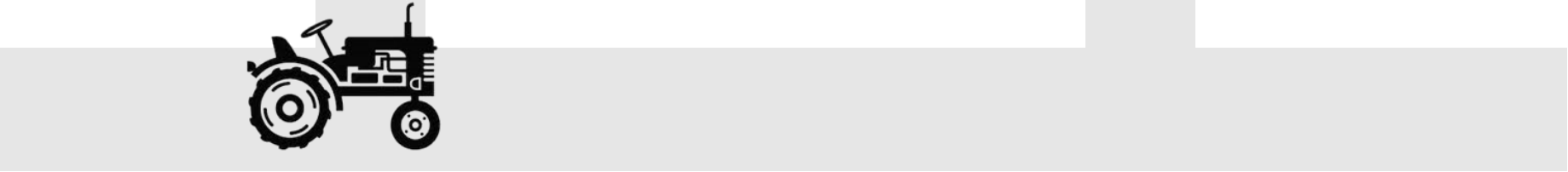


Field



Robot

Robot

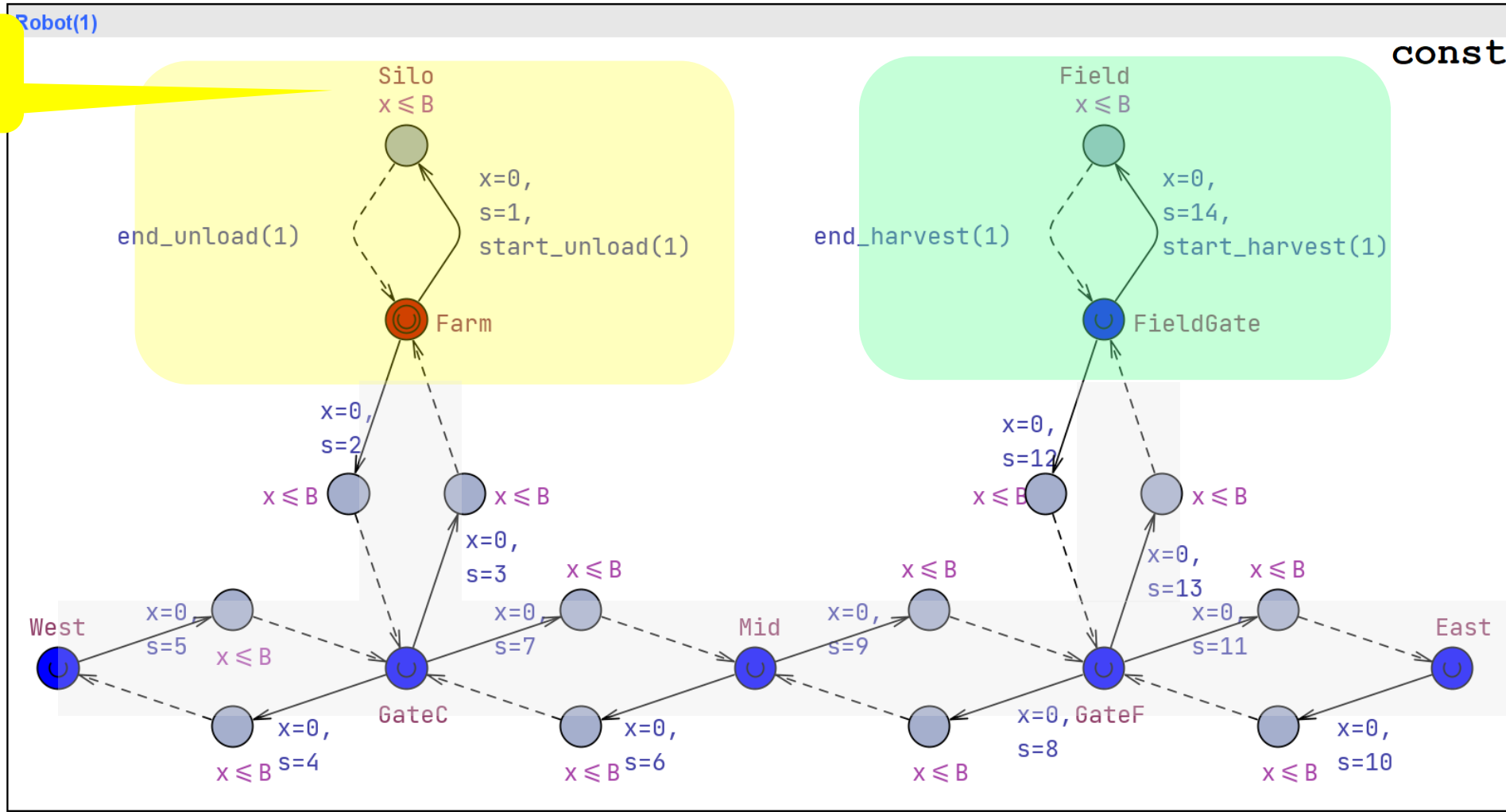


Road



# Smart Farming – Timed Automata

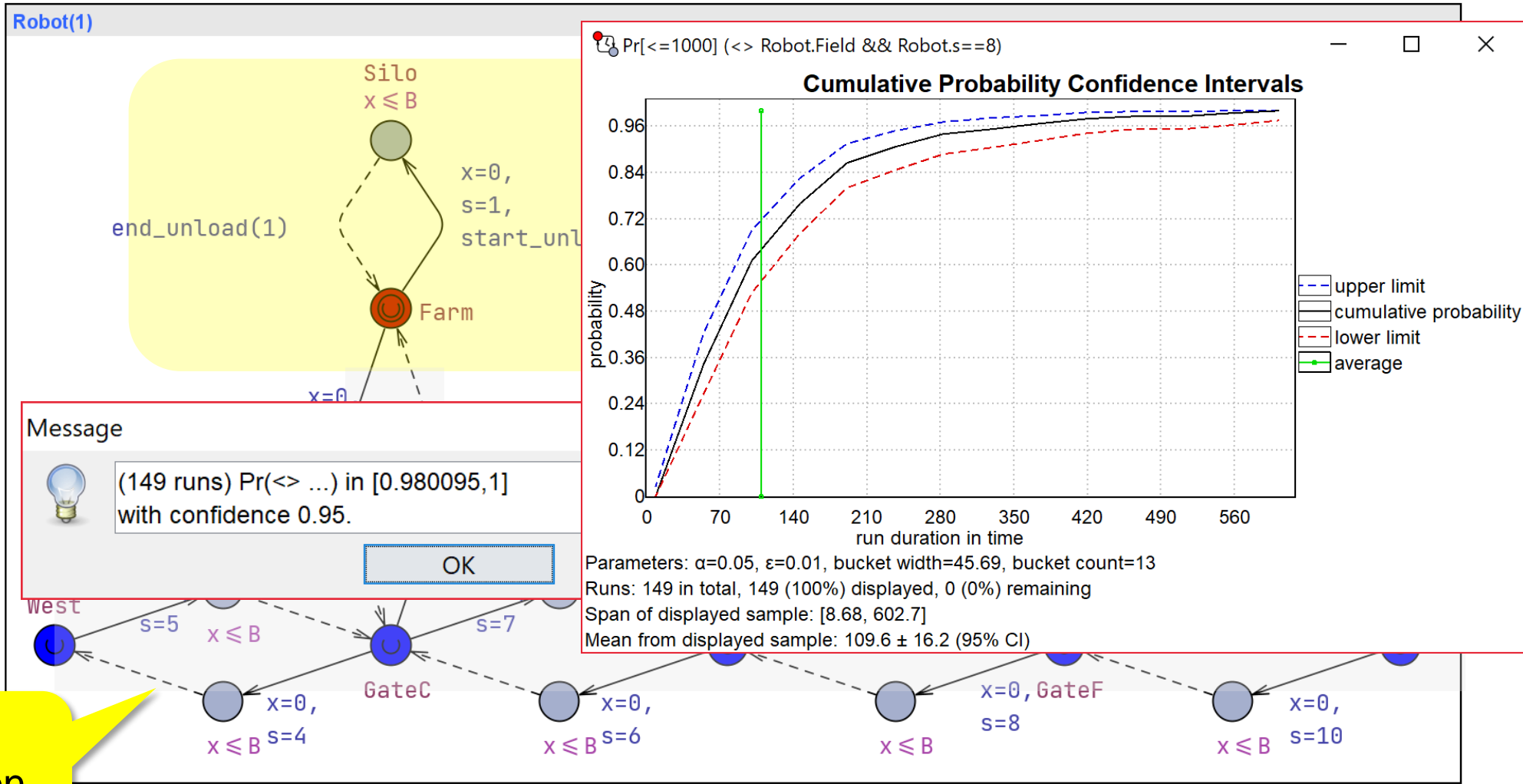
Invariant



```
const int B=5;
clock x;
int s=1;
```

$E \leftrightarrow \text{Robot}(1).\text{Field} \ \&\& \ \text{Robot}(1).s == 14$

# Smart Farming – Stochastic Timed Automata



$B=5;$   
 $lock\ x;$   
 $t\ s=1;$

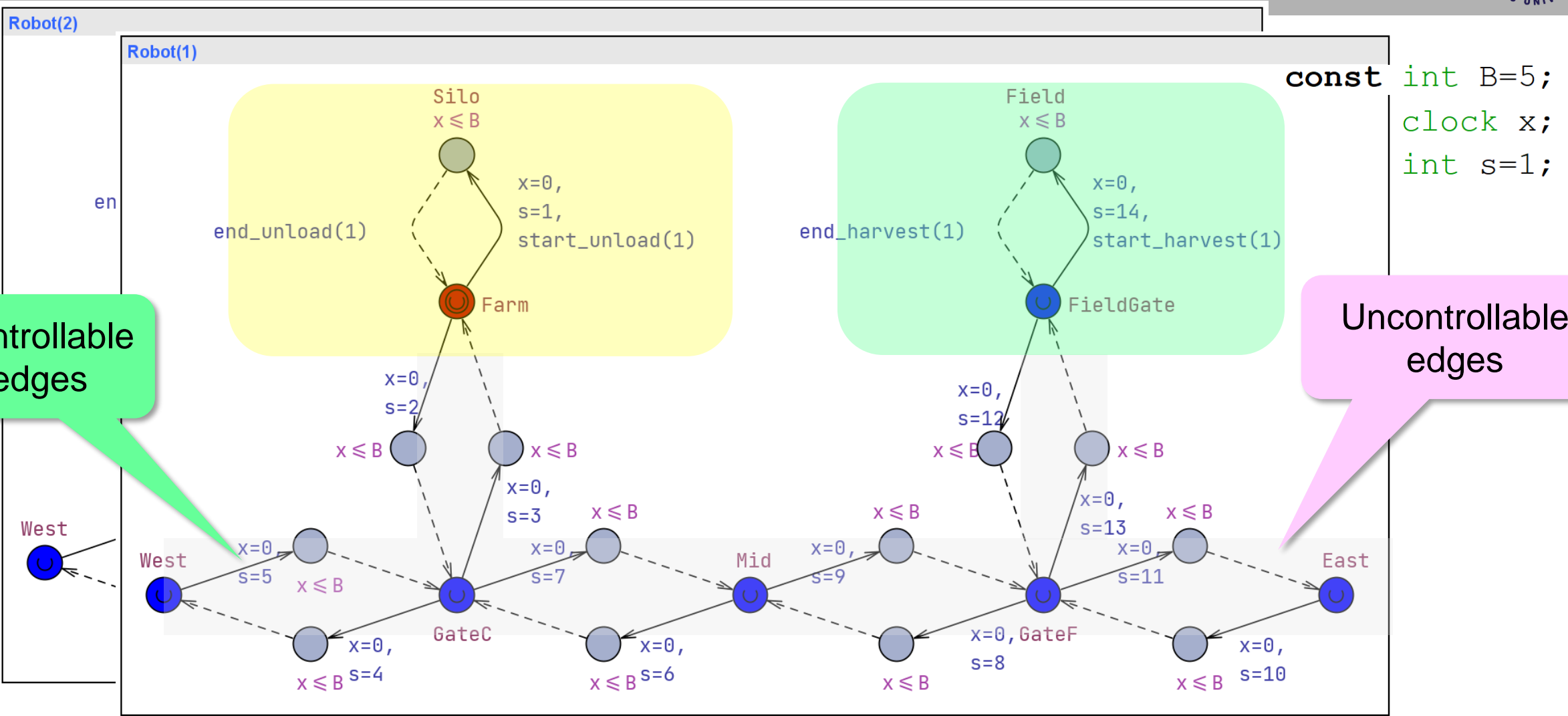
Information

Uniform distribution  $[0, B]$

$Pr[\leq 1000] (\langle \rangle Robot(1).Field \ \&\& \ Robot(1).s==14)$

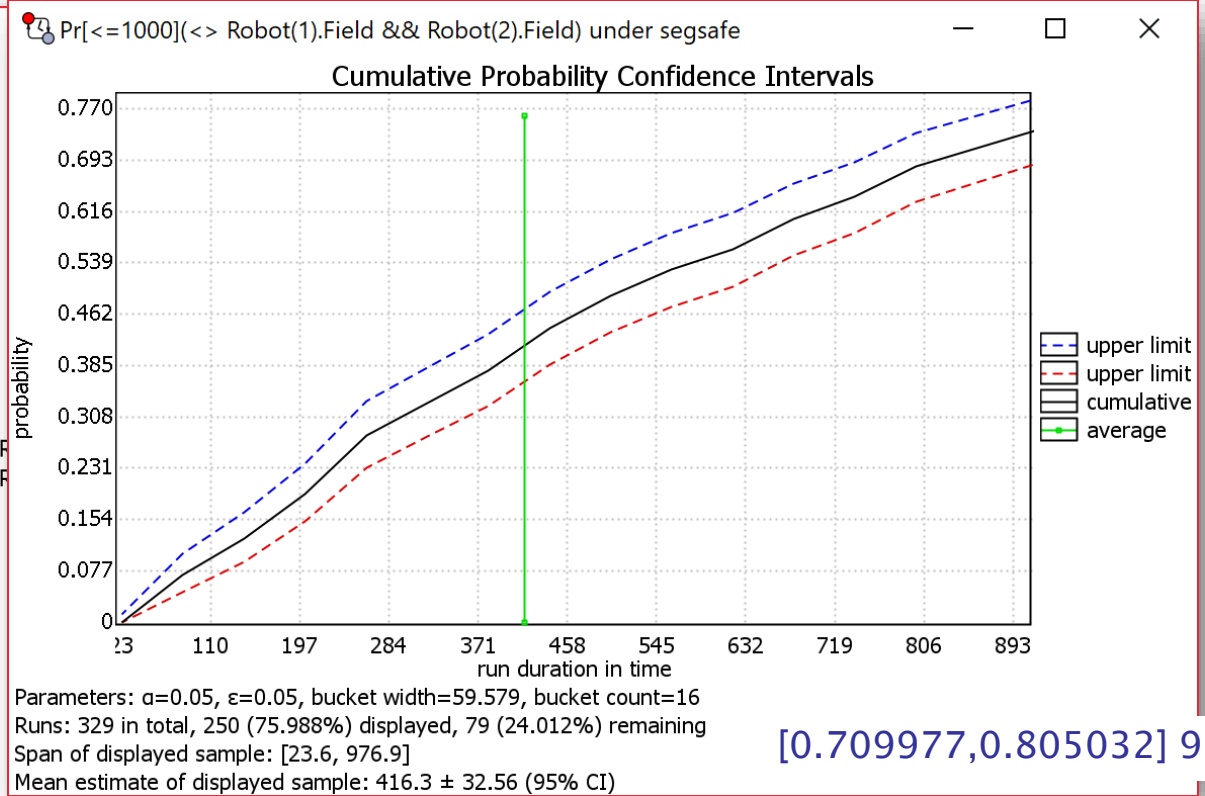
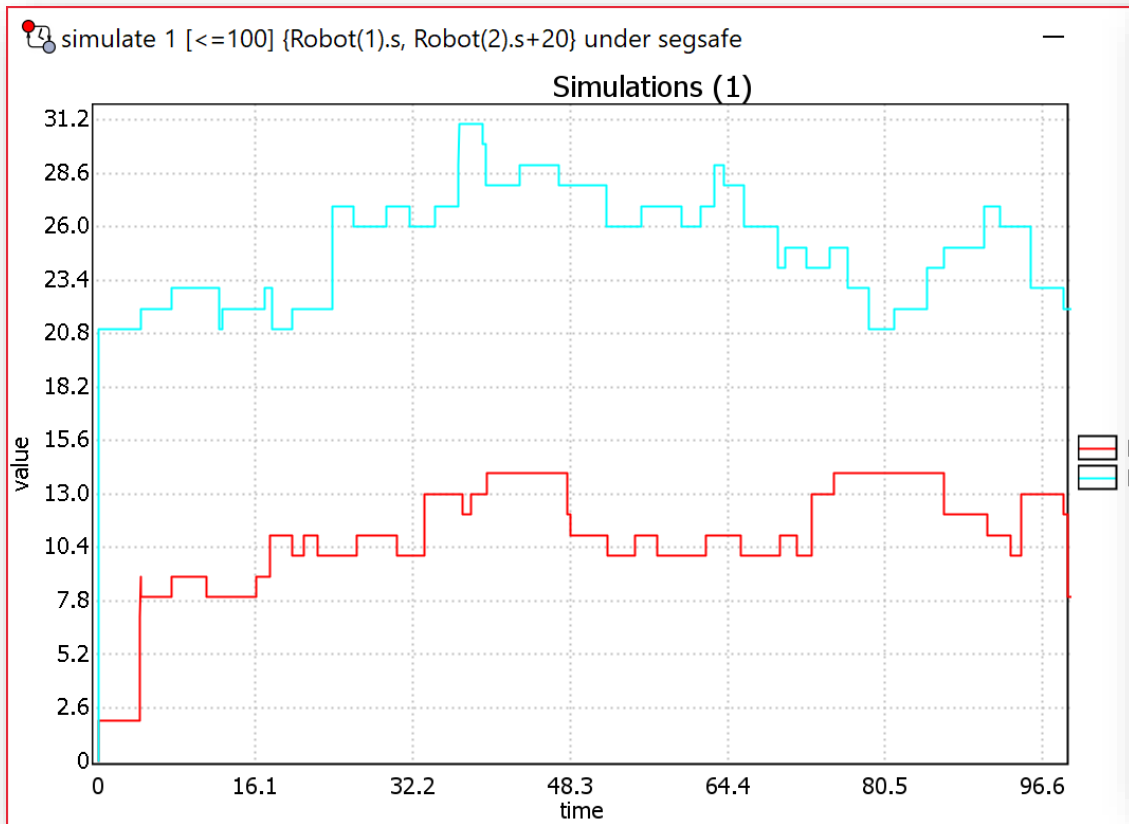


# Smart Farming - Timed Game



strategy segsafe = control: A[] ! ( Robot(1).s>1 && Robot(1).s<14 && Robot(1).s==Robot(2).s)

# Smart Farming – Timed Games

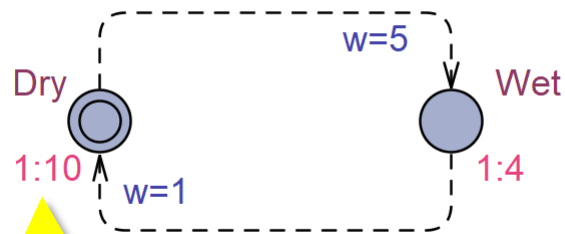


[0.709977, 0.805032] 95%

simulate 1 [ $\leq 100$ ] {Robot(1).s, Robot(2).s+20} under segsafe  
 $E \langle \rangle$  Robot(1).Field && Robot(2).Field under segsafe  
 $E \langle \rangle$  Robot(1).s > 1 && Robot(1).s < 14 && Robot(1).s == Robot(2).s under segsafe  
 Pr[ $\leq 1000$ ] ( $\langle \rangle$  Robot(1).Field && Robot(2).Field == 14) under segsafe

# Smart Farming – Stochastic & Hybrid Stuff

Rain



Field



$$f' = (((F-f)/(0.5*F))*w)*scale - (harvesting[1]*f*(1 - (1-f)/F) + harvesting[2]*f*(1 - (1-f)/F))$$

Store



$$c' = storing[1]*ld[1] + storing[2]*ld[2]$$

ODEs

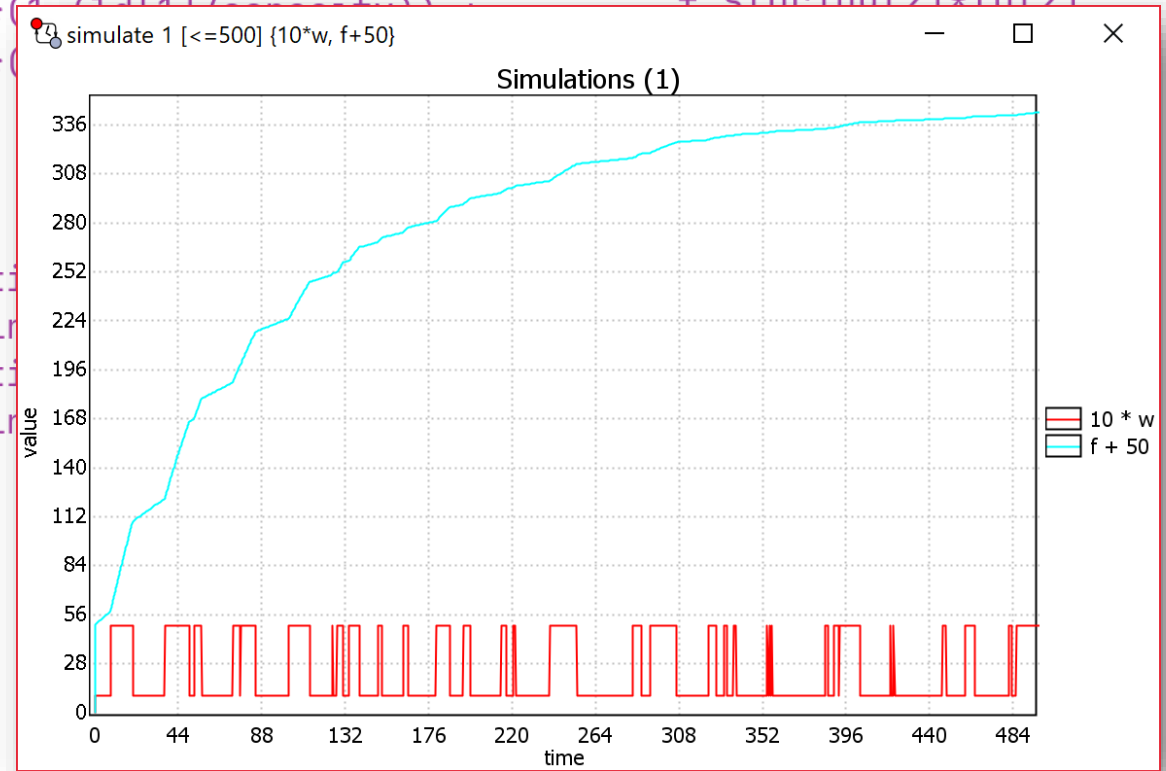
Rates of exponential distributions

Load



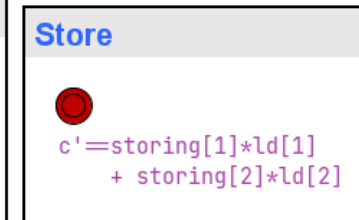
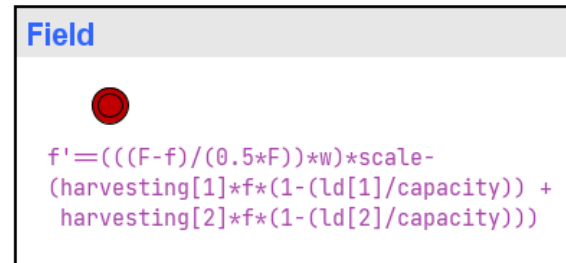
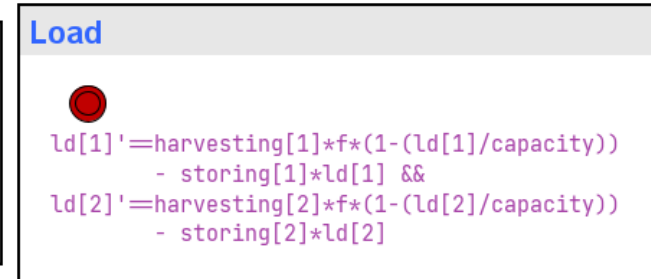
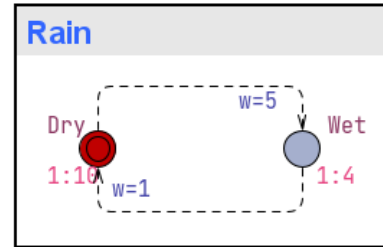
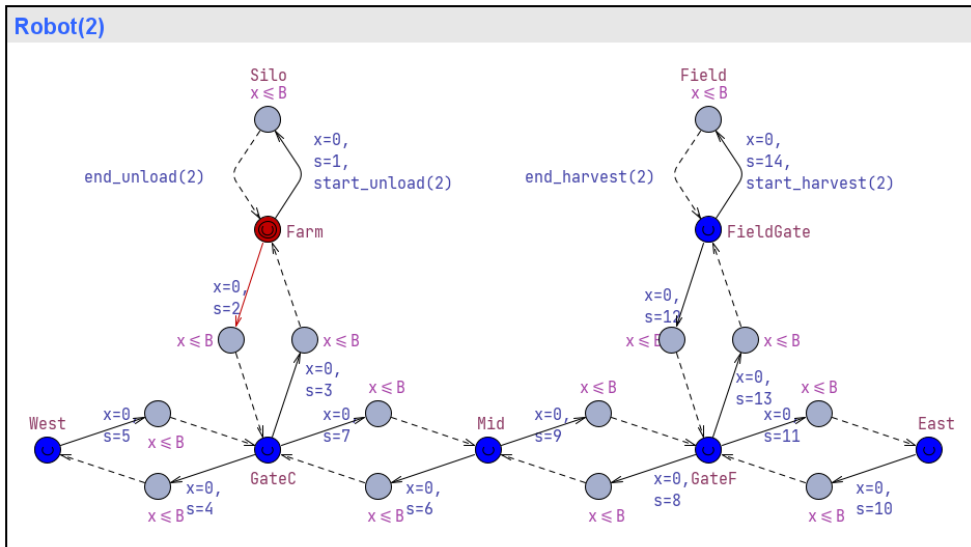
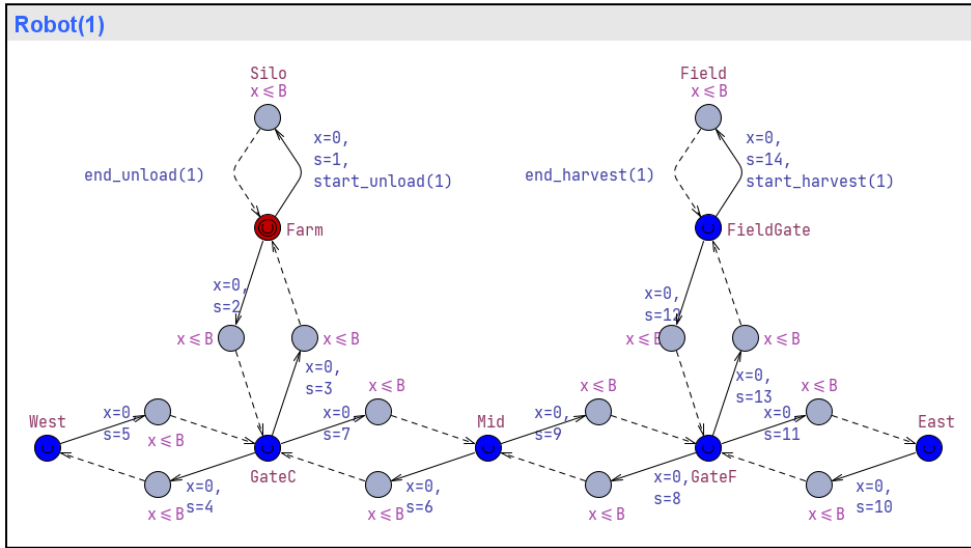
$$ld[1]' = harvesting[1] - storing[1]*ld[1]$$

$$ld[2]' = harvesting[2] - storing[2]*ld[2]$$



```
const double capacity = 40.0;
hybrid clock ld[id_t];
hybrid clock f, c;
```

# Smart Farming – Complete Model



```
typedef int[1,2] id_t;
const int B=5;
int w=1;
clock t;

const double capacity = 40.0;
hybrid clock ld[id_t];
hybrid clock f, c;
```

```
bool harvesting[id_t] = {false, false};
bool storing[id_t] = {false, false};

bool capacity_check(id_t rid) {
    if(ld[rid] ≥ capacity)
        return 0;
    else
        return 1;
}

void start_unload(id_t rid) {
    storing[rid] = true;
}
```

# Farming Benchmark – in Stratego

eval\_farm.xml - UPPAAL

File Edit View Tools Options Help

Editor Simulator ConcreteSimulator Verifier

Transition chooser

0.0 2.0 4.0 6.0 8.0

Urgent 0 Expand

Reset Next Shrink

Globals

```
w = 5.0
harvesting = {0.0,0.0}
  [1] = 0.0
  [2] = 0.0
storing = {0.0,1.0}
#t(0) = 0.0
#time = 50.9
t = 50.9
ld = {10.036268339675237,
  [1] = 10.0362683396752
  [2] = 14.4169373466321}
f = 33.890448475993956
c = 41.05864741200095
```

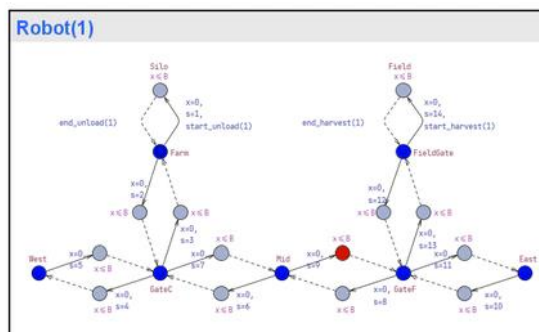
Robot(1)

```
rid = 1.0
s = 9.0
x = 0.0
```

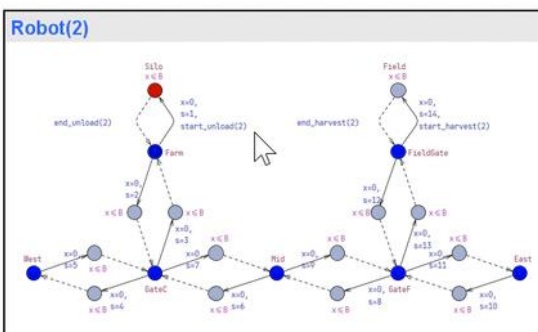
Robot(2)

```
rid = 2.0
s = 1.0
x = 0.2
```

Robot(1)



Robot(2)



Load

```
ld[1]'=harvesting[1]+f*(1-(ld[1]/capacity)) - storing[1]*ld[1] &&
ld[2]'=harvesting[2]+f*(1-(ld[2]/capacity)) - storing[2]*ld[2]
```

Field

```
f'=((f-f/0.5f)*w)+scale-
(harvesting[1]+f*(1-(ld[1]/capacity)) +
harvesting[2]+f*(1-(ld[2]/capacity)))
```

Store

```
c'=storing[1]*ld[1] + storing[2]*ld[2]
```

Simulation Trace

Robot(2)

(-, Silo, Dry, -, -, -)

Rain

(-, Silo, Wet, -, -, -)

Robot(1)

(Mid, Silo, Wet, -, -, -)

Robot(1)

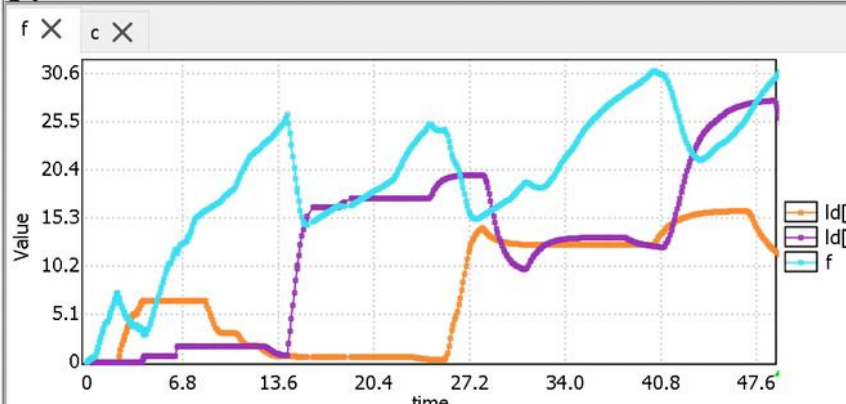
(-, Silo, Wet, -, -, -)

Prev 49.361 Next

Replay Stochastic

Slow Fast

f × c ×

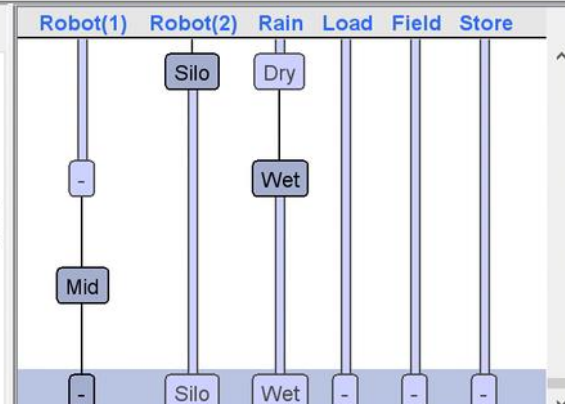


Value

time

ld[1] ld[2] f

Robot(1) Robot(2) Rain Load Field Store



# Farming Benchmark – in Stratego



eval\_farm\_Kim.xml - UPPAAL

File Edit View Tools Options Help

Editor Simulator ConcreteSimulator Verifier

Overview

```
// Two Robots Two way Road
strategy segsafe = control: A[] ! ( Robot(1).s>1 && Robot(2).s<14 && Robot(1).s==Robot(2).s)
strategy opt_harvest = maxE(c) [<=1000] {Robot(1).location, Robot(2).location, Rain.location} → {ld[1],ld[2],f}: < t ≥ 1000 under segsafe
E<> Robot(1).s>1 && Robot(1).s<14 && Robot(1).s==Robot(2).s
E<> (Robot(1).s>1 && Robot(2).s<14 && Robot(1).s==Robot(2).s) under segsafe

simulate 1 [<=1000] {Robot(1).s, Robot(2).s+10}
simulate 1 [<=500] {Robot(1).s, Robot(2).s+20} under segsafe
simulate 10 [<=500] {Robot(1).s, Robot(2).s+20} under opt_harvest

simulate 1 [<=500] {ld[1],ld[2],f} under opt_harvest
```

Query

```
simulate 1 [<=500] {Robot(1).s, Robot(2).s+20} under segsafe
```

Comment

Status

Property is satisfied.

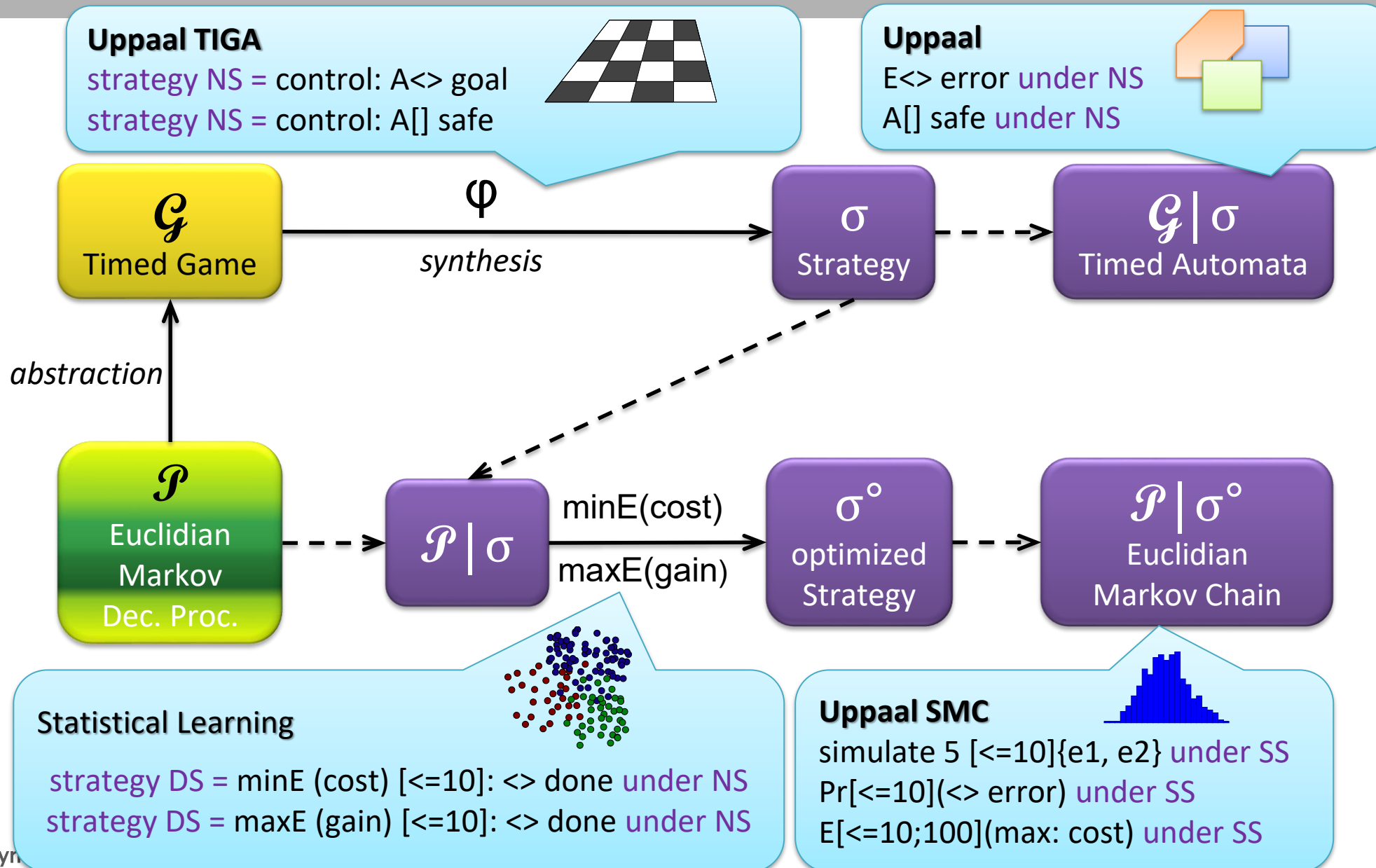
Check  
Insert above  
Insert below  
Remove  
Comments  
Clear results

www.facebook.com • now ^  
Facebook  
Wang Yi har slået en opdateri  
(71 andre nye notifikationer)

www.facebook.com • now  
Facebook



# Workflow under UPPAAL Stratego



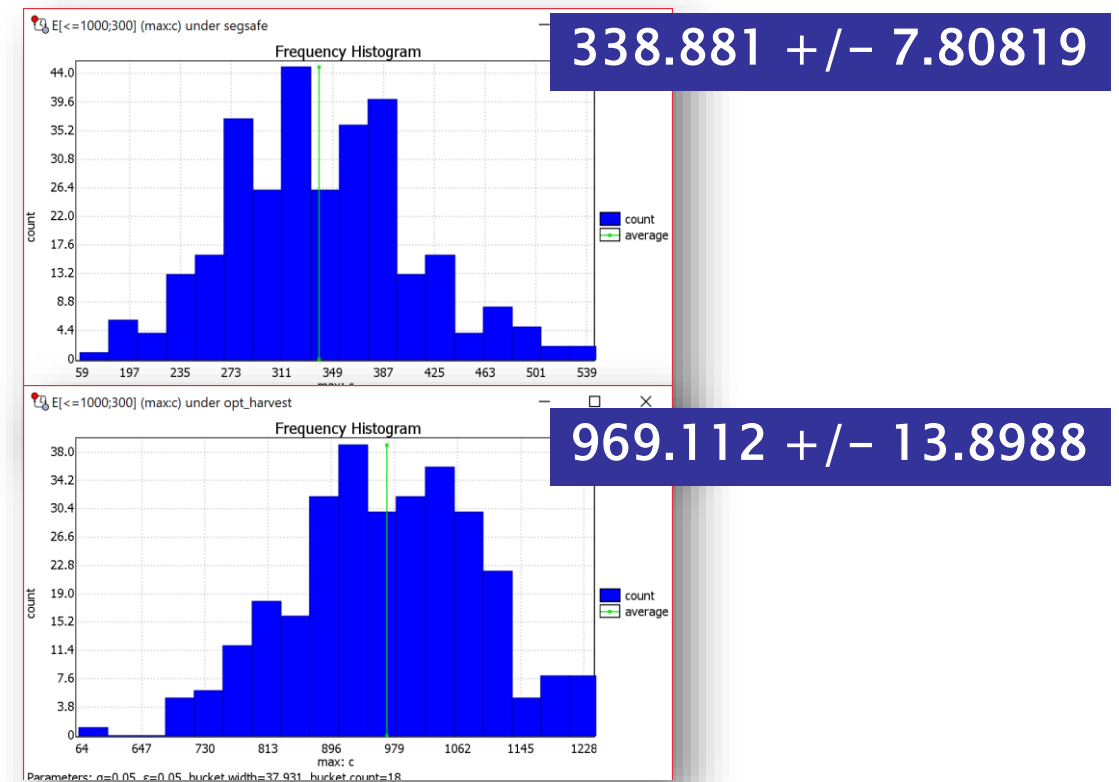
# Smart Farming

Q1: strategy **segsafe** = control:  $A[] ! ( \text{Robot}(1).s > 1 \ \&\& \ \text{Robot}(1).s < 14 \ \&\& \ \text{Robot}(1).s == \text{Robot}(2).s )$

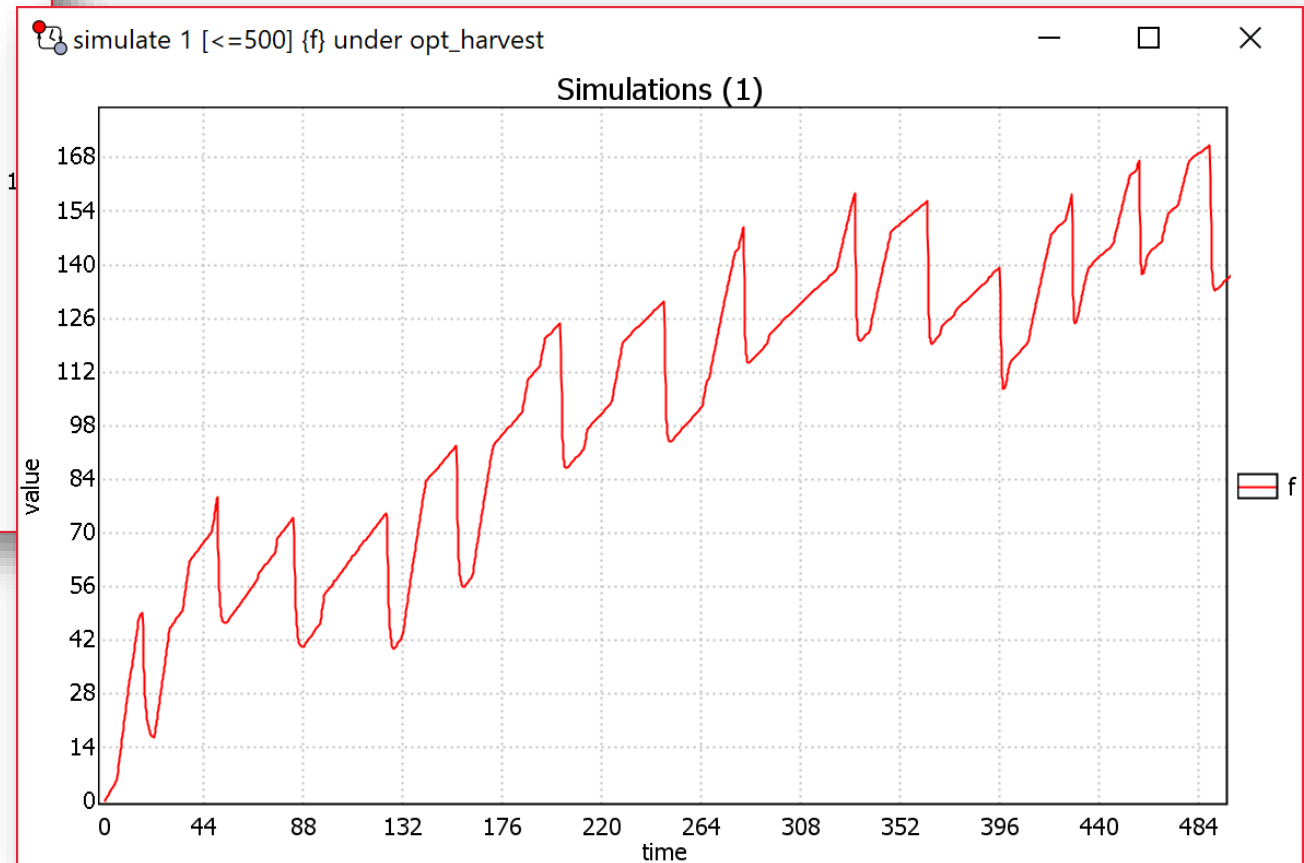
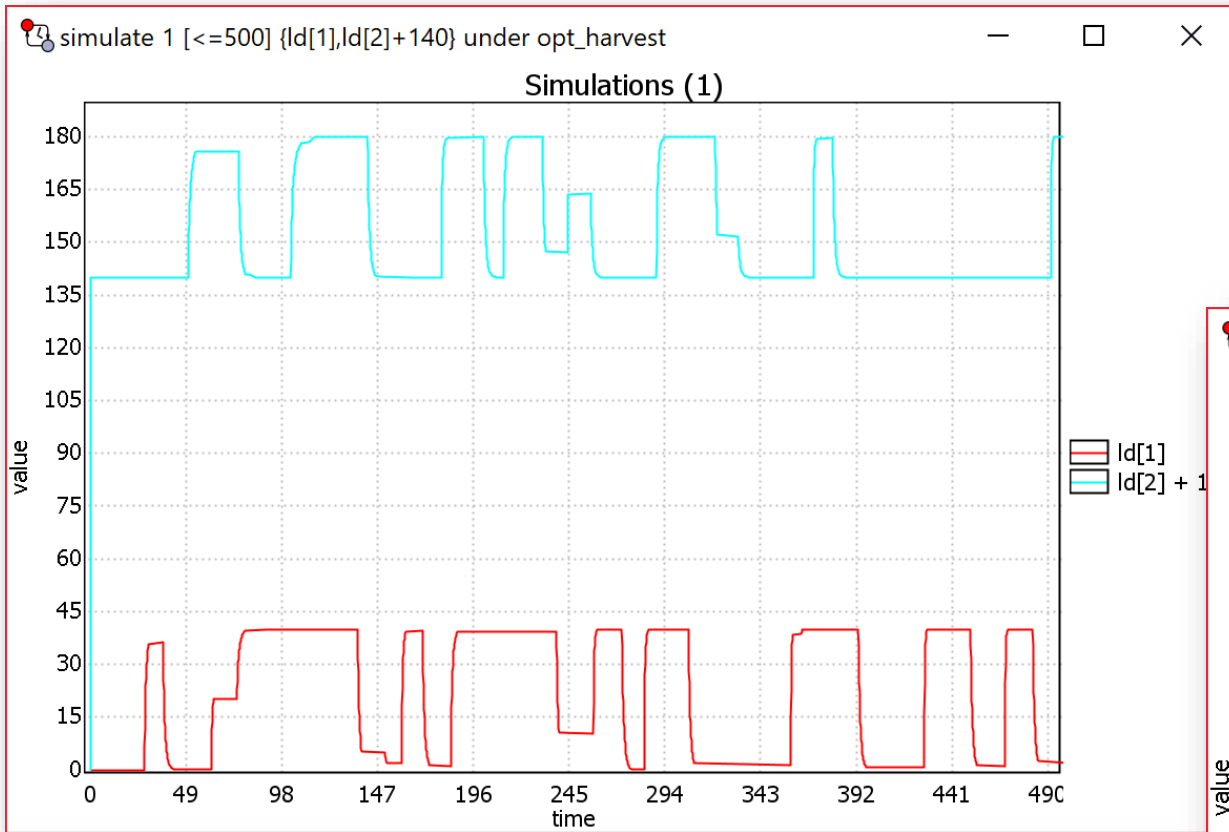
Q2: strategy **opt\_harvest** =  
 $\text{maxE}(c) [ \leq 1000 ] \{ \text{Robot}(1).location, \text{Robot}(2).location, \text{Rain}.location \} \rightarrow \{ l_1, l_2, t \}$   
 $\langle \rangle t \geq 1000$  under **segsafe**

Q3:  $E[ \leq 1000; 300 ]$  (**max:c**) under **segsafe**

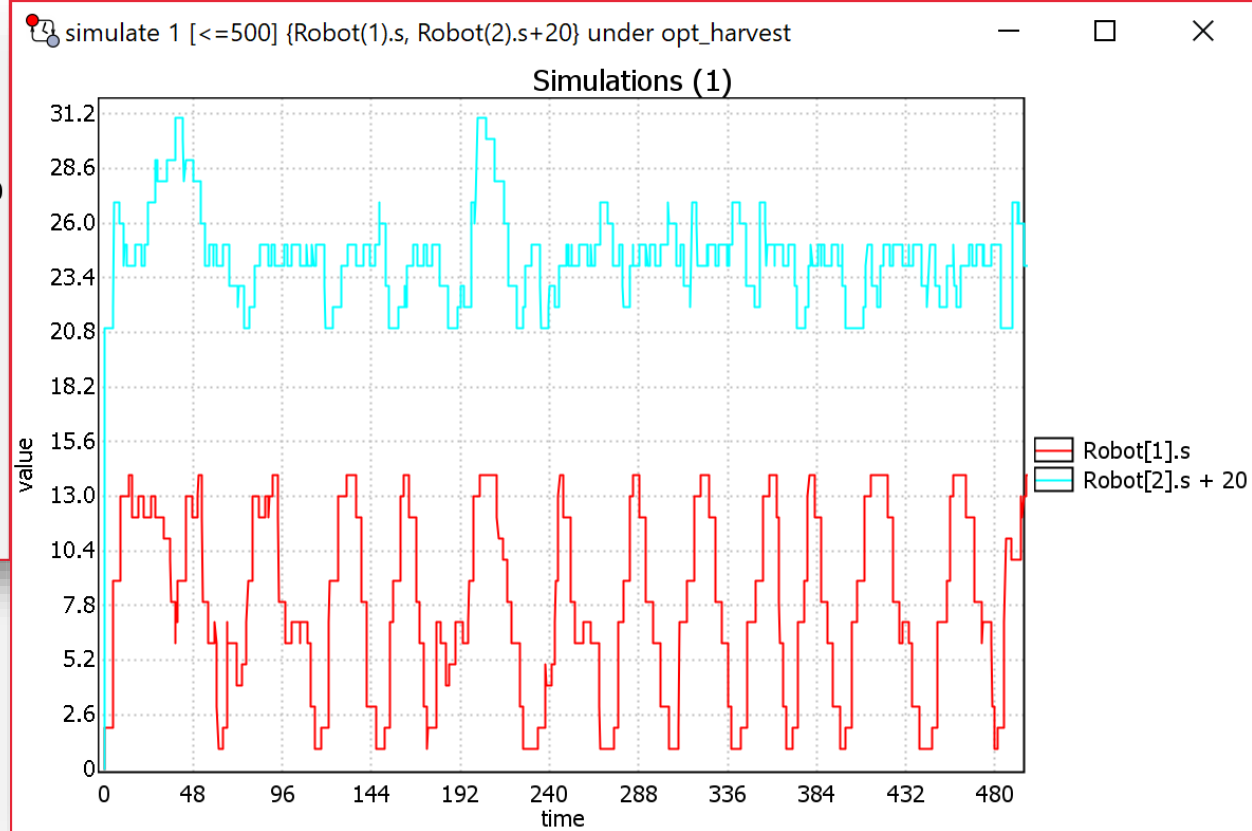
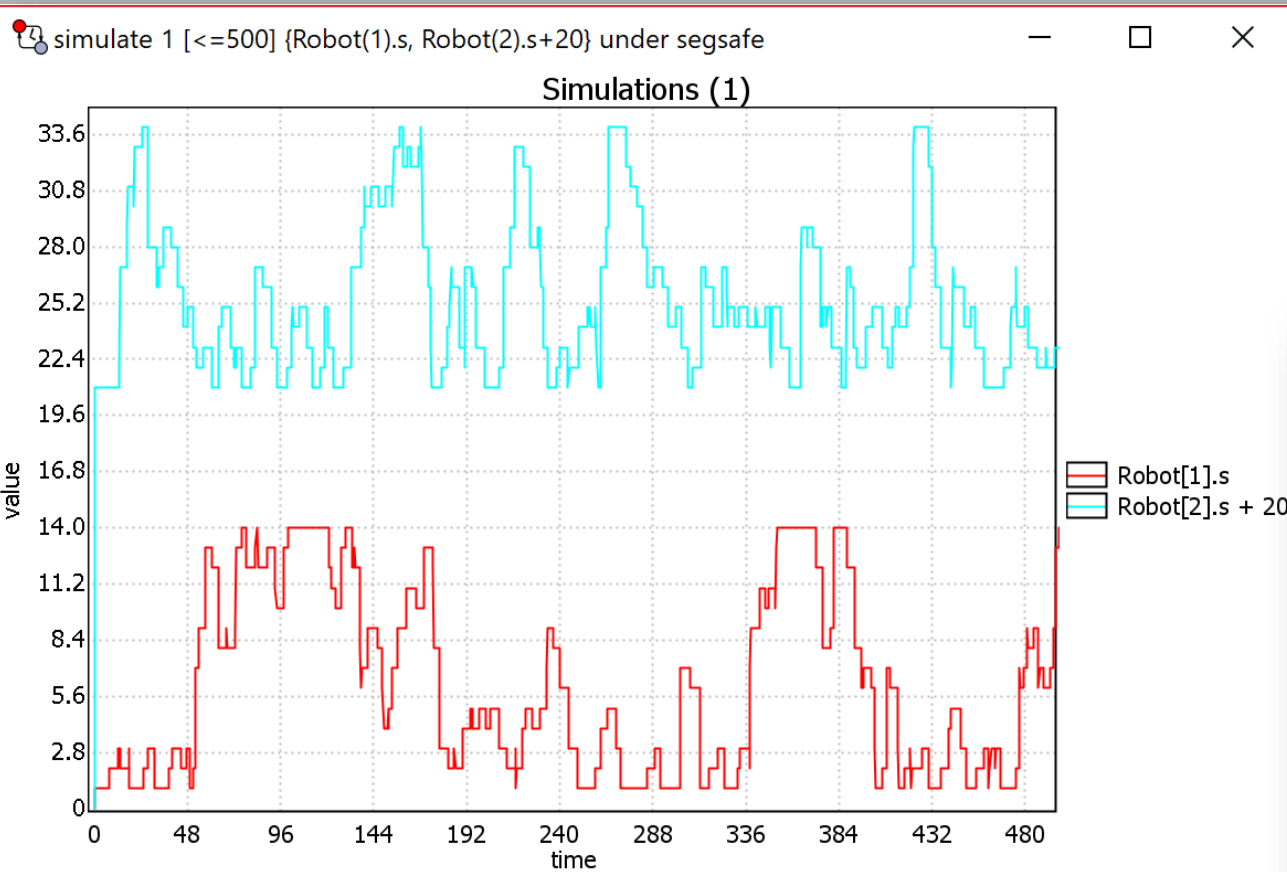
Q4:  $E[ \leq 1000; 300 ]$  (**max:c**) under **opt\_harvest**



# ld[1], ld[2] – f



# Robots Movement



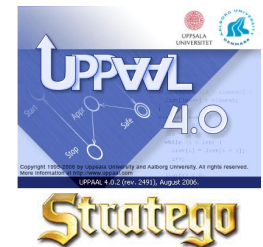
# Better Learning



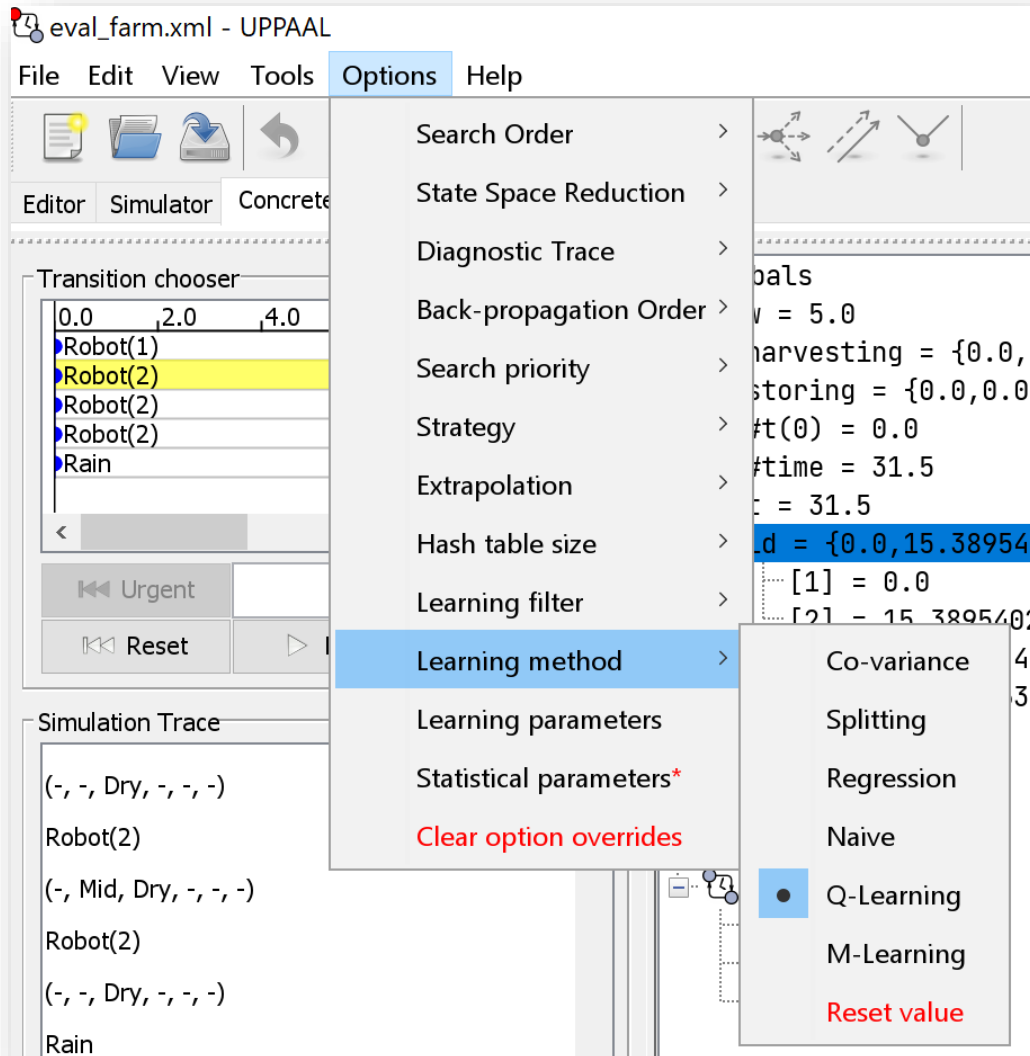
AALBORG UNIVERSITET



VILLUM FONDEN



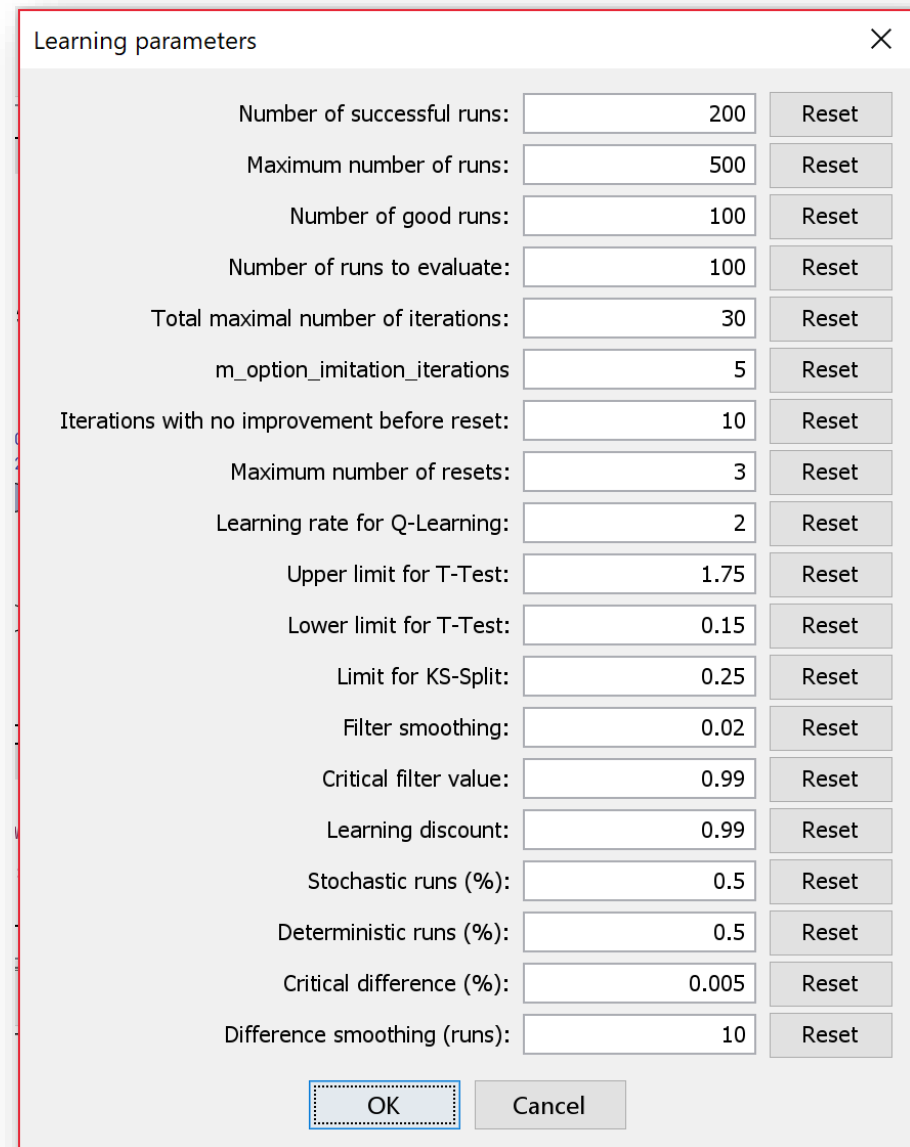
# Learning Methods & Parameters



The screenshot shows the UPPAAL software interface. The 'Options' menu is open, and 'Learning method' is selected. The sub-menu shows the following options:

- Co-variance
- Splitting
- Regression
- Naive
- Q-Learning** (selected)
- M-Learning
- Reset value

The background shows the 'Transition chooser' with a list of transitions: Robot(1), Robot(2), Robot(2), Robot(2), and Rain. The 'Robot(2)' transitions are highlighted in yellow. The 'Simulation Trace' shows a sequence of states: (-, -, Dry, -, -, -), Robot(2), (-, Mid, Dry, -, -, -), Robot(2), (-, -, Dry, -, -, -), and Rain.



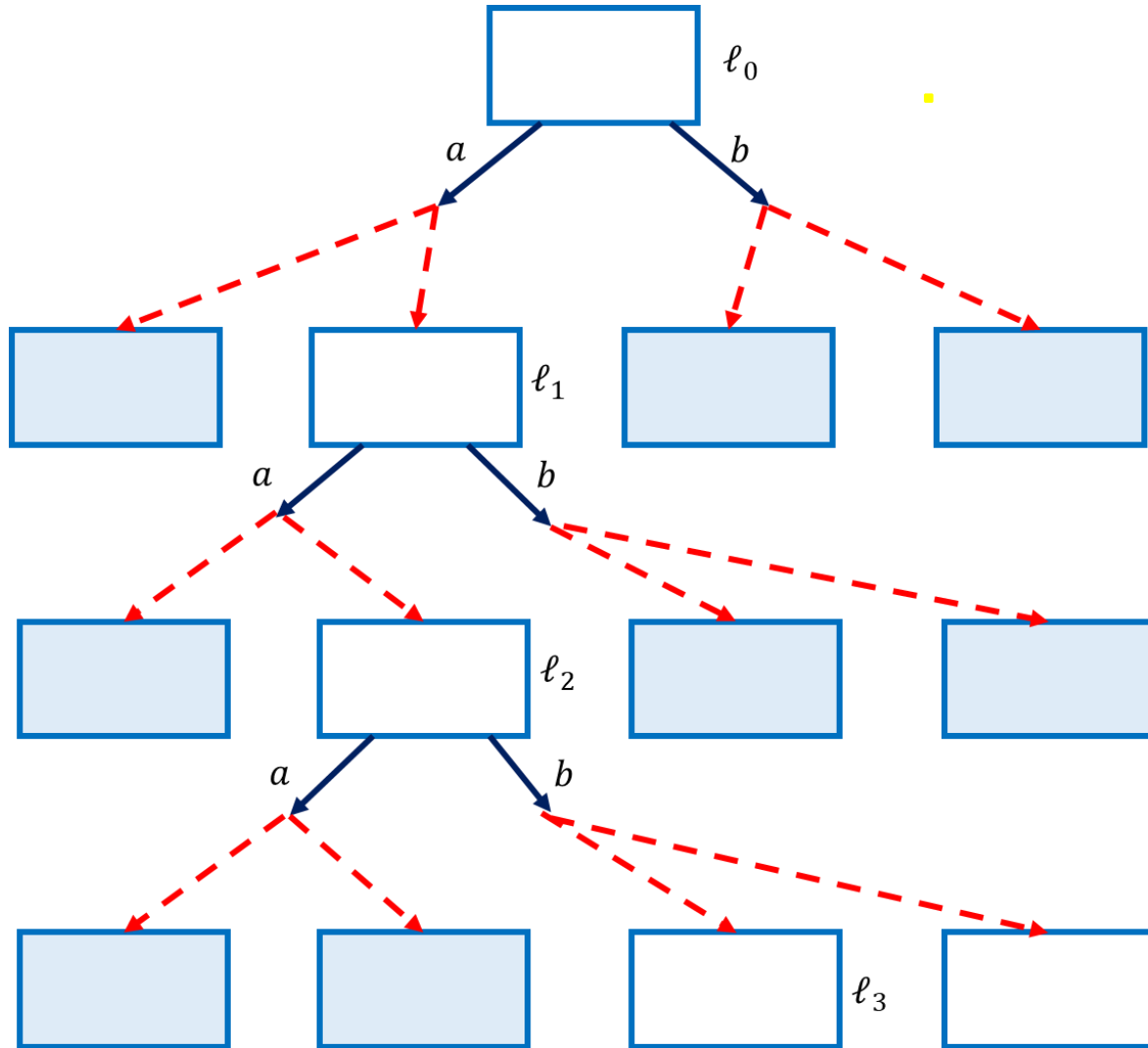
The 'Learning parameters' dialog box is shown, containing the following parameters and their values:

Parameter	Value	Reset
Number of successful runs:	200	Reset
Maximum number of runs:	500	Reset
Number of good runs:	100	Reset
Number of runs to evaluate:	100	Reset
Total maximal number of iterations:	30	Reset
m_option_imitation_iterations	5	Reset
Iterations with no improvement before reset:	10	Reset
Maximum number of resets:	3	Reset
Learning rate for Q-Learning:	2	Reset
Upper limit for T-Test:	1.75	Reset
Lower limit for T-Test:	0.15	Reset
Limit for KS-Split:	0.25	Reset
Filter smoothing:	0.02	Reset
Critical filter value:	0.99	Reset
Learning discount:	0.99	Reset
Stochastic runs (%):	0.5	Reset
Deterministic runs (%):	0.5	Reset
Critical difference (%):	0.005	Reset
Difference smoothing (runs):	10	Reset

Buttons: OK, Cancel



# Euclidean MDP



- States  $\mathcal{S} \subseteq \mathbb{R}^k$
- **Act** is a finite set of actions
- Initial state  $s_0 \in \mathcal{S}$
- Next state density function  $T: \mathcal{S} \times \text{Act} \rightarrow (\mathcal{S} \rightarrow \mathbb{R}_{\geq 0})$
- Cost-function  $C: \mathcal{S} \times \text{Act} \times \mathcal{S} \rightarrow \mathbb{R}$
- Goal states  $G \subseteq \mathcal{S}$

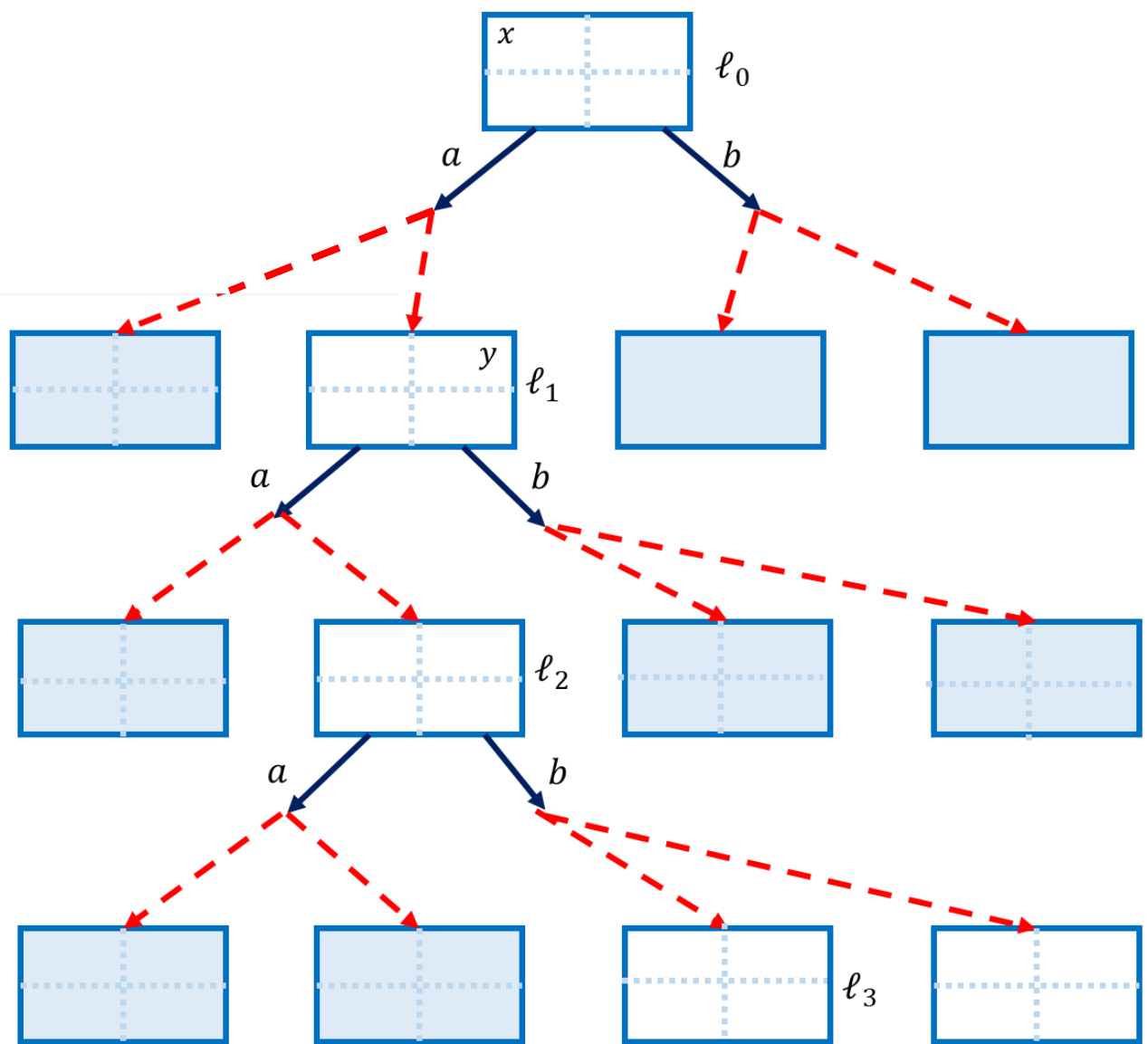
Strategy  $\sigma: \mathcal{S} \rightarrow (\text{Act} \rightarrow [0,1])$

$\mathbb{E}_\sigma(G, s)$ :

Expected cost of reaching  $G$  under  $\sigma$ .

Find  $\sigma^*$  st.  $\mathbb{E}_{\sigma^*}(G, s) = \inf_{\sigma} \mathbb{E}_{\sigma}(G, s)$

# Partitioning (IMDP)



- Finite partition  $A$  (respecting  $G$ )
- Next state transition-function  

$$T_A: A \times Act \times A \rightarrow [0, 1] \times [0, 1]$$
- Cost-function  

$$C_A: A \times Act \times A \rightarrow \mathbb{R} \times \mathbb{R}$$

$$\mathbb{E}_A^{min}(G, x) \quad \mathbb{E}_A^{max}(G, x)$$

**Theorem**  

$$\mathbb{E}_A^{min}(G, [s]) \leq \inf_{\sigma} \mathbb{E}_{\sigma}(G, s) \leq \mathbb{E}_A^{max}(G, [s])$$

**Theorem**  
**(Bounded horizon, continuous cost & transition)**  
 Let  $A_0 \sqsubseteq A_1 \sqsubseteq \dots \sqsubseteq A_i \sqsubseteq \dots$  a be refining sequence  
 “of arbitrary precision”. Then  

$$\inf_{i \rightarrow \infty} \mathbb{E}_{A_i}^{min}(G, [s]_{A_i}) = \inf_{\sigma} \mathbb{E}_{\sigma}(G, s) = \inf_{i \rightarrow \infty} \mathbb{E}_{A_i}^{max}(G, [s]_{A_i})$$

# Q- & M-Learning

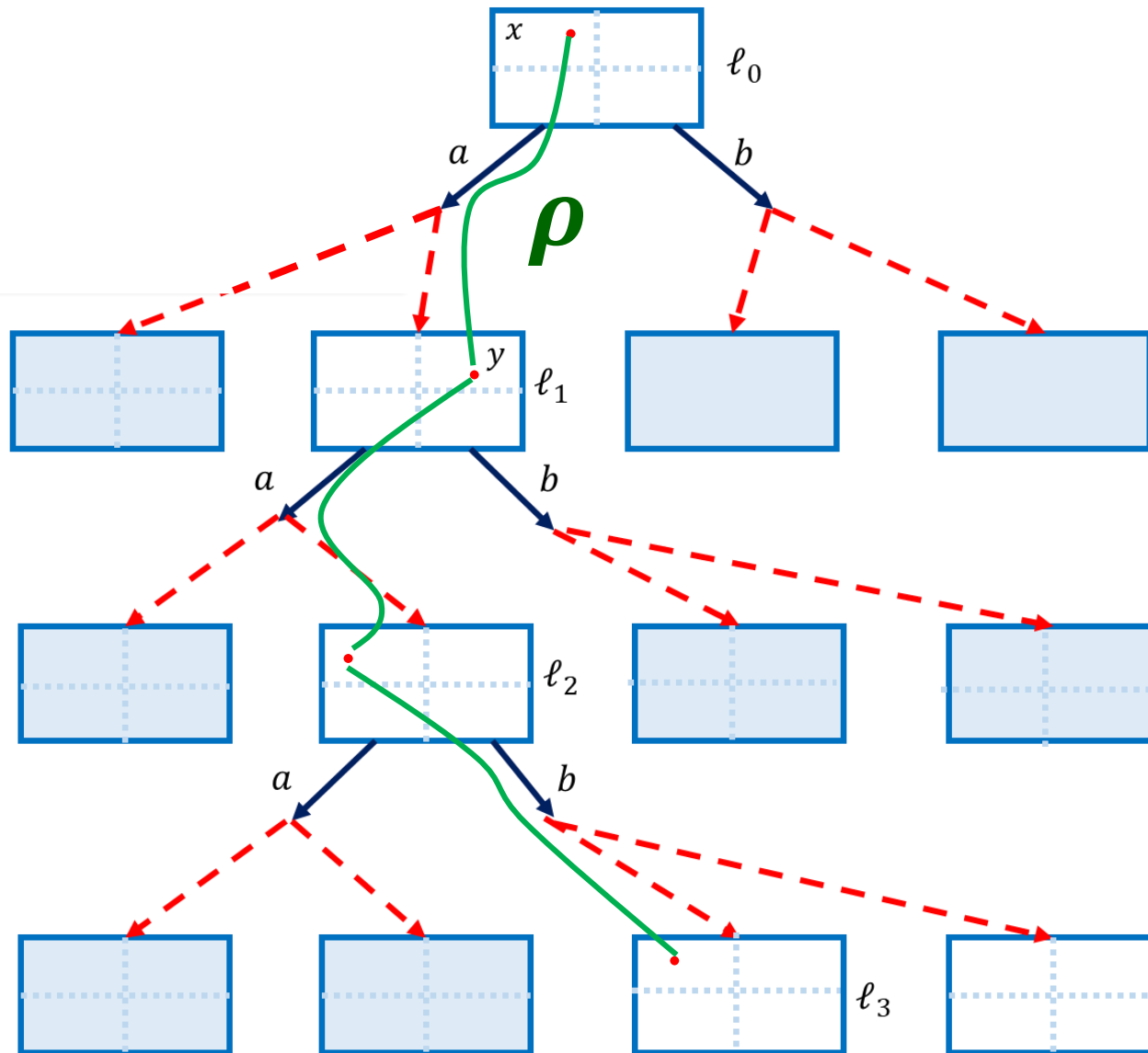
$$Q_a^{\ell,x}, Q_b^{\ell,x} : \mathbb{R}_{\geq 0} \quad x \in \mathcal{P}$$

$$\#(\ell, x) : \mathbb{N}$$

$$\#((\ell, x), \alpha, \{\ell', y\}) : \mathbb{N}$$

$$C((\ell, x), \alpha, (\ell', y)) : \mathbb{R}_{\geq 0}$$

$$x, y \in \mathcal{P}, \alpha \in \{a, b\}$$



## REPEATEDLY

1. Generate a random run  $\rho$ 
  - using  $Q_a^{\ell,x}$  and  $Q_b^{\ell,x}$  as weights for randomly choosing between  $a$  and  $b$  in  $\ell$
  - Using model for stochastic choice.

# Q- and M-Learning

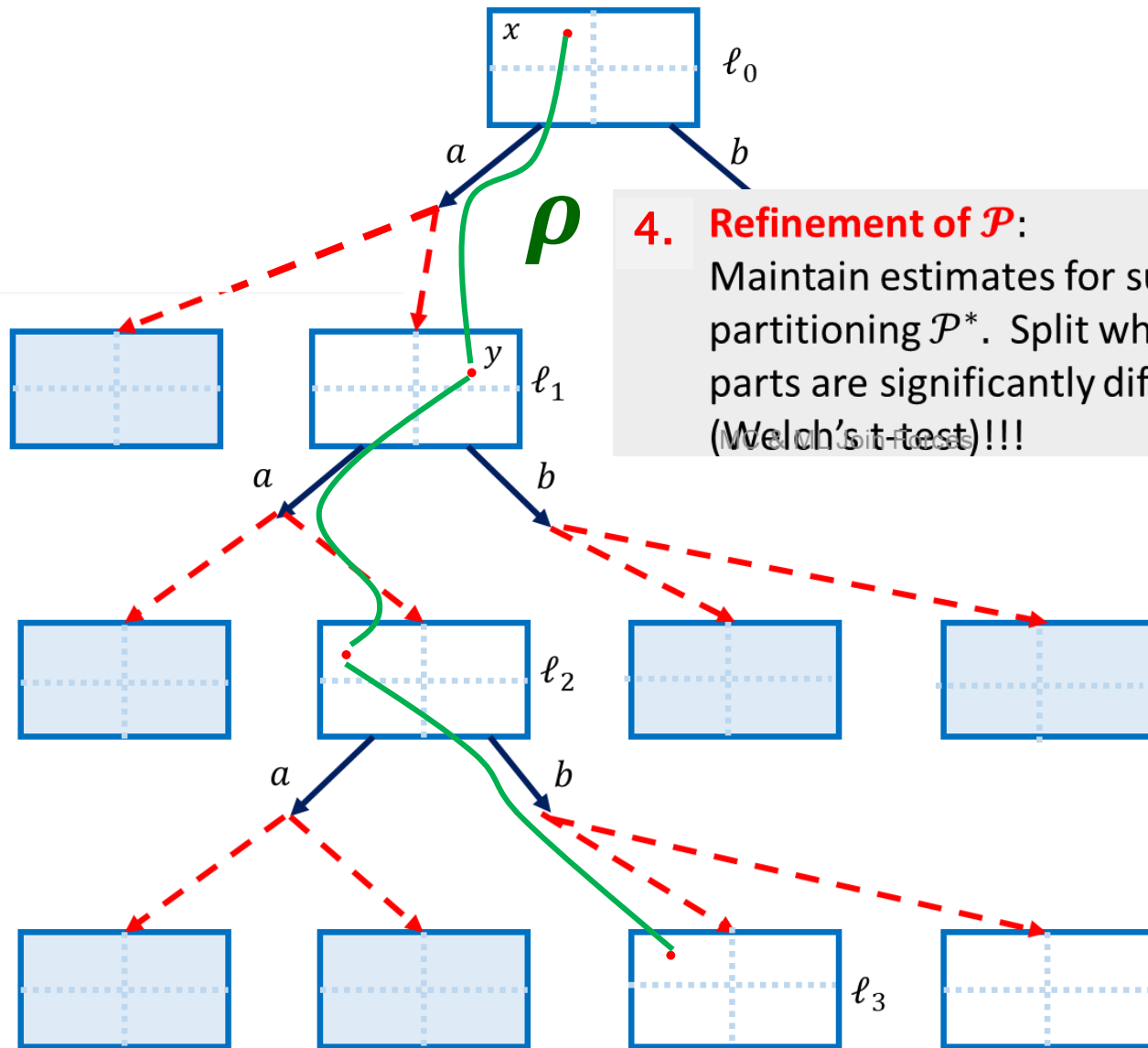
$$Q_a^{\ell,x}, Q_b^{\ell,x} : \mathbb{R}_{\geq 0} \quad x \in \mathcal{P}$$

$$\#(\ell, x) : \mathbb{N}$$

$$\#((\ell, x), \alpha, \{\ell', y\}) : \mathbb{N}$$

$$C((\ell, x), \alpha, (\ell', y)) : \mathbb{R}_{\geq 0}$$

$$x, y \in \mathcal{P}, \alpha \in \{a, b\}$$



**4. Refinement of  $\mathcal{P}$ :**  
 Maintain estimates for sub-partitioning  $\mathcal{P}^*$ . Split when sub-parts are significantly different (Welch's t test)!!!

## REPEATEDLY

1. Generate a random run  $\rho$ 
  - using  $Q_a^{\ell,x}$  and  $Q_b^{\ell,x}$  as weights for randomly choosing between  $a$  and  $b$  in  $\ell$
  - Using model for stochastic choice.

2. Update
  - $\#(\ell, x)$
  - $\#((\ell, x), \alpha, (\ell', y))$
  - $C((\ell, x), \alpha, (\ell', y))$
 along the run

3. Backwards update  $Q_a^{\ell,x}$  and  $Q_b^{\ell,x}$

$$Q_a^{\ell_0,x} = \left(1 - \frac{1}{\#(\ell_0, x)}\right) \cdot Q_a^{\ell_0,x} + \frac{1}{\#(\ell_0, x)} \cdot (C(s_0, a, s_1) + \min_{\alpha} Q_{\alpha}^{\ell_1,y})$$

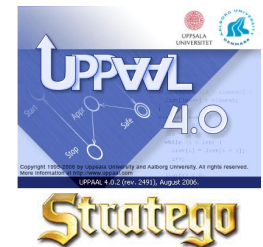
# Approximations



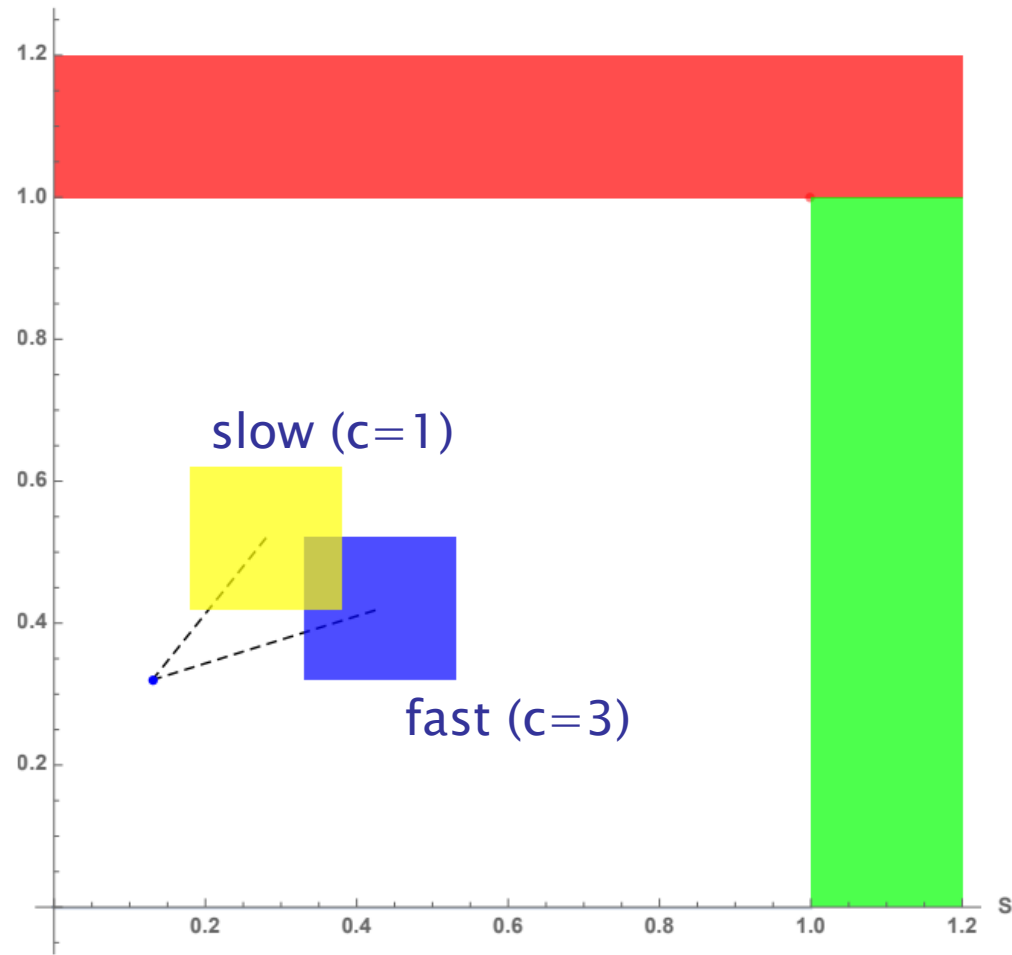
AALBORG UNIVERSITET



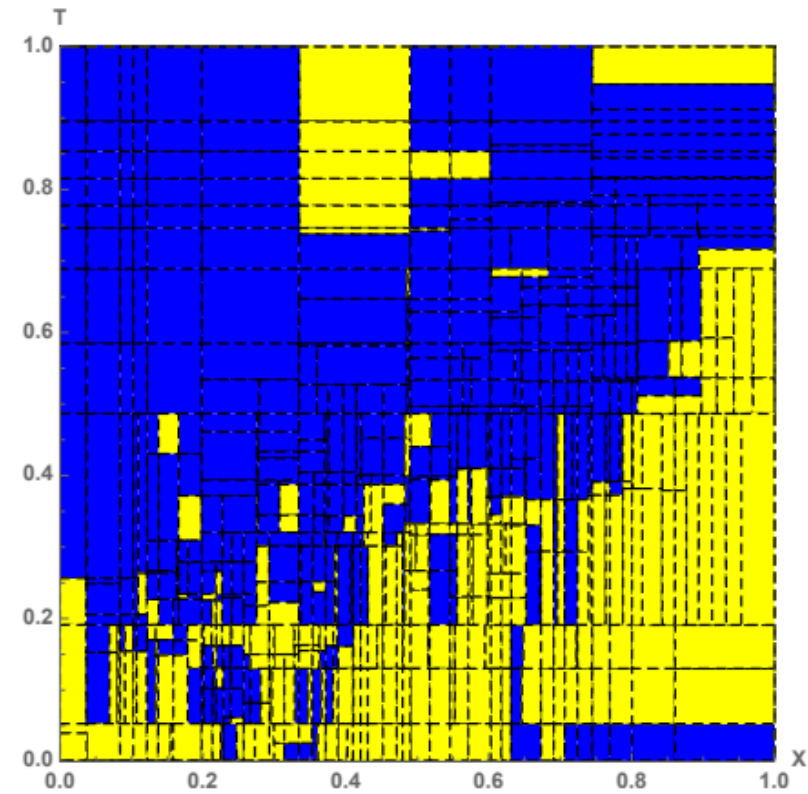
VILLUM FONDEN



# Semi-Random Walk



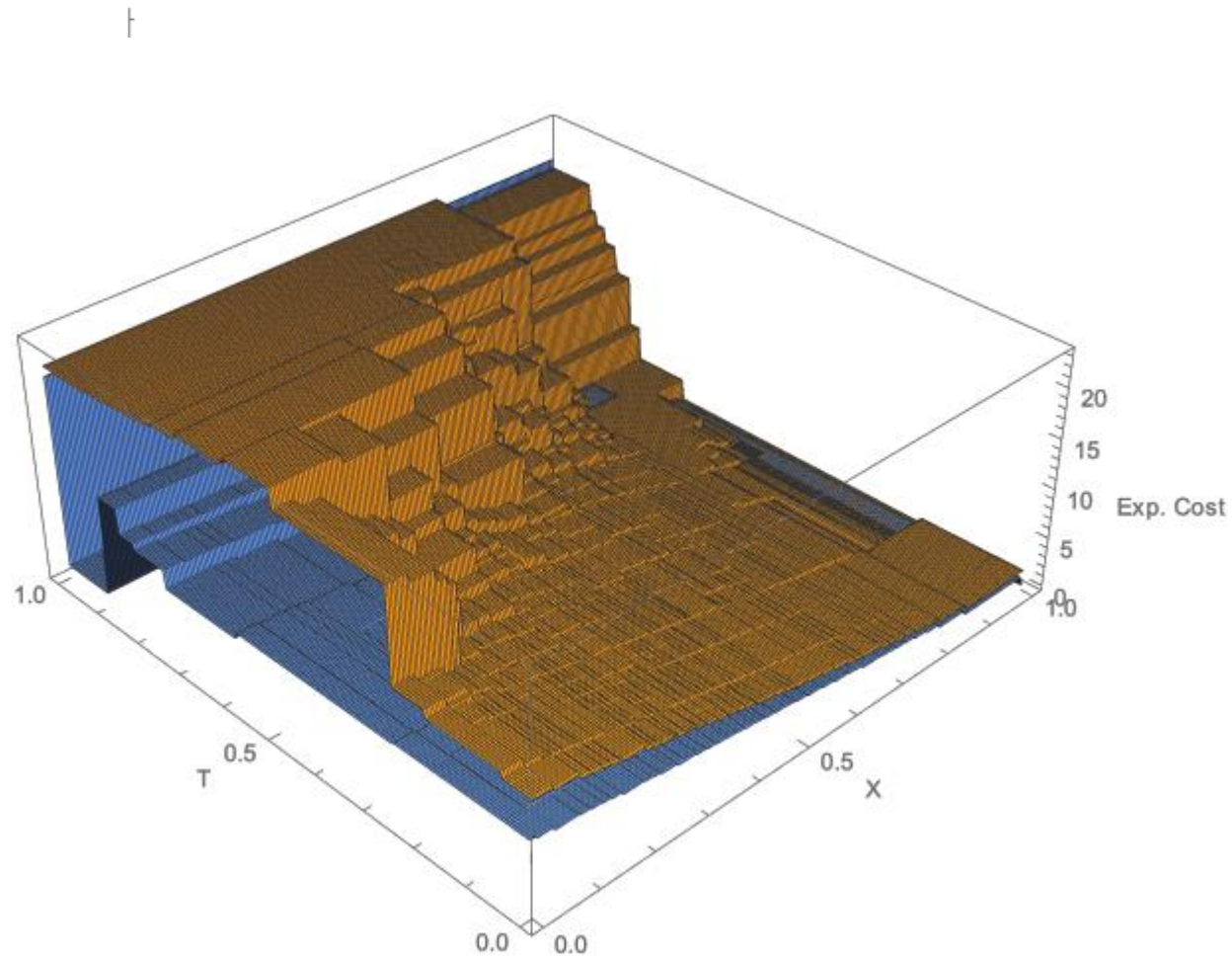
EMDP



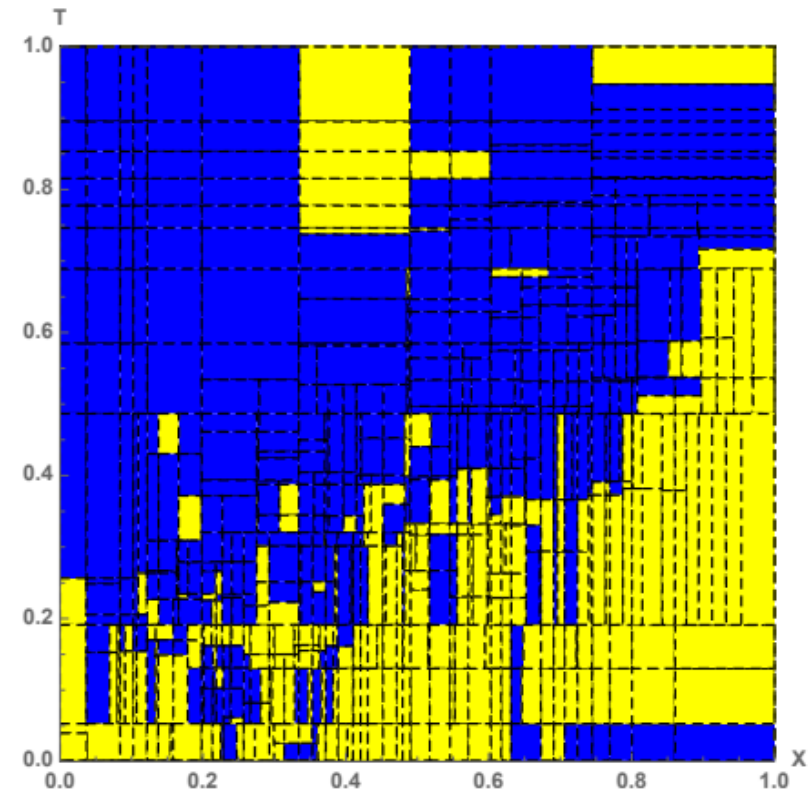
Strategy learned by STRATEGO



# Semi-Random Walk

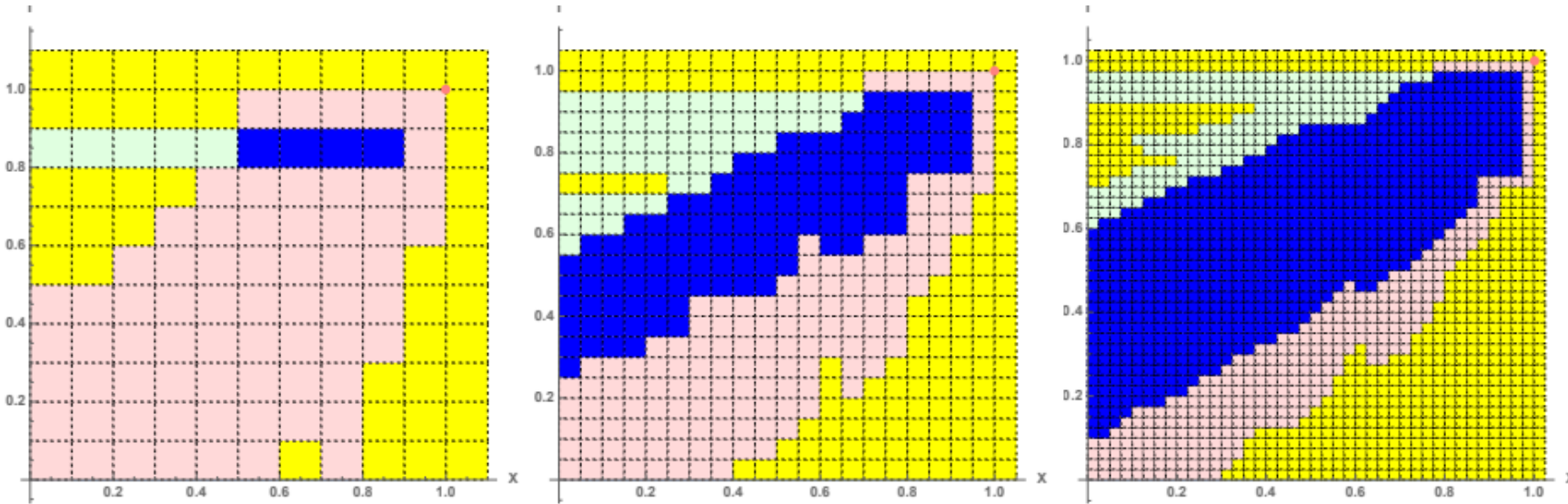


Lower/Upper Cost-bound



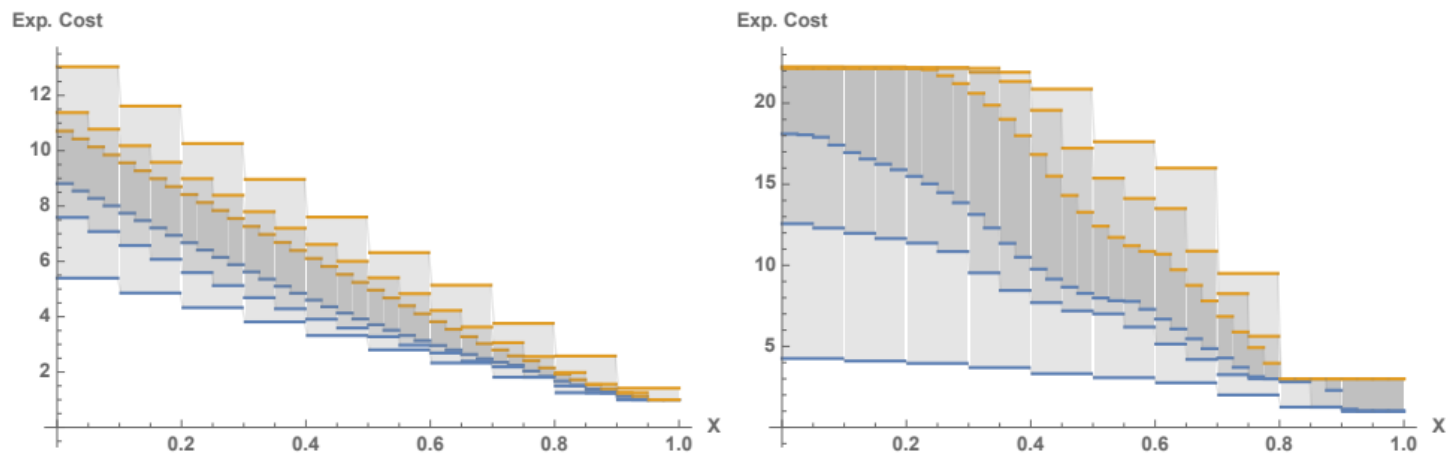
Strategy learned by STRATEGO

# Strategies from Successive Partitioning

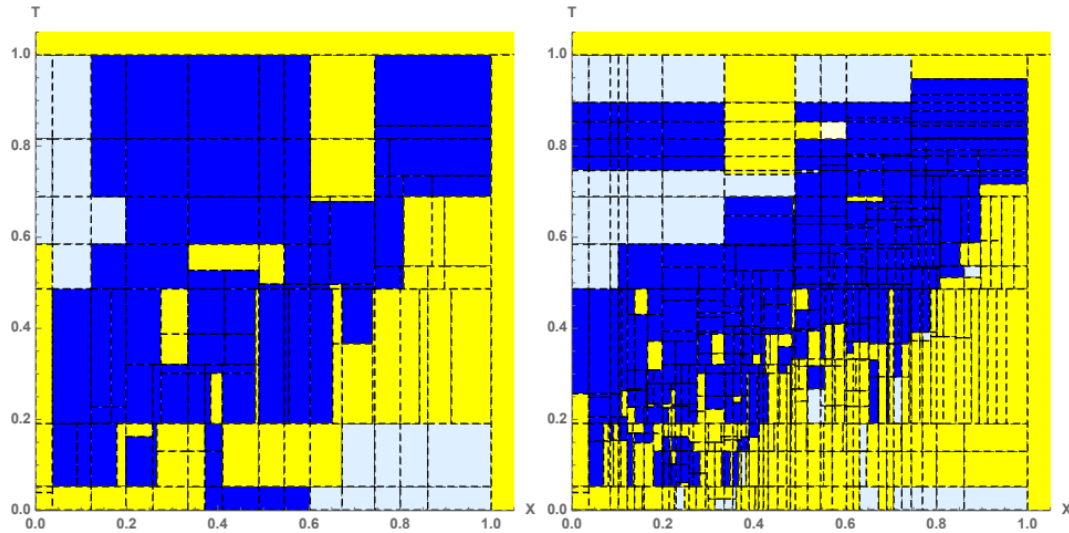


Obtained from lower and upper expected cost approximations

L/U expected costs  
for  $t=0.0$ ,  $t=0.7$



# Strategies from Learning



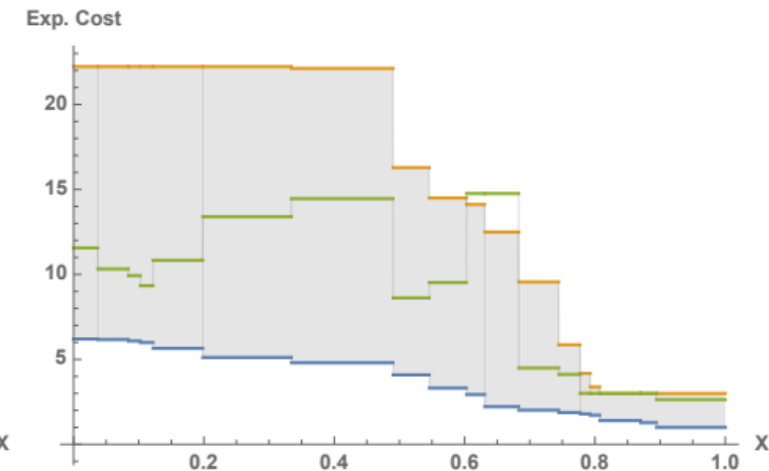
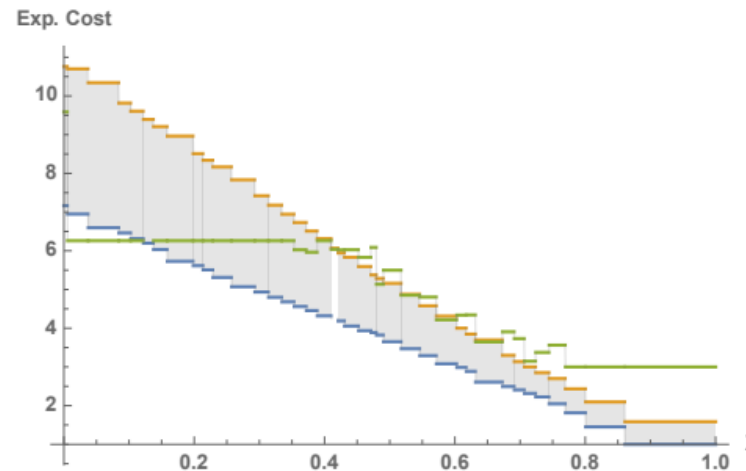
k=27

k=205

Expected cost functions for k=205  
Along  $t=0.0$ ,  $t=0.7$ .

**Yellow:** Upper expected cost  
**Blue:** Lower expected cost  
**Green:** Learned expected cost

**CHALLENGE**  
Prove convergence  
of Q-learning for  
EMDPs



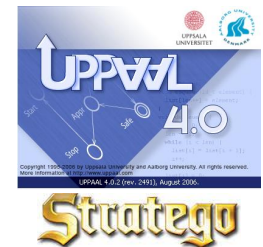
# Compact Strategies



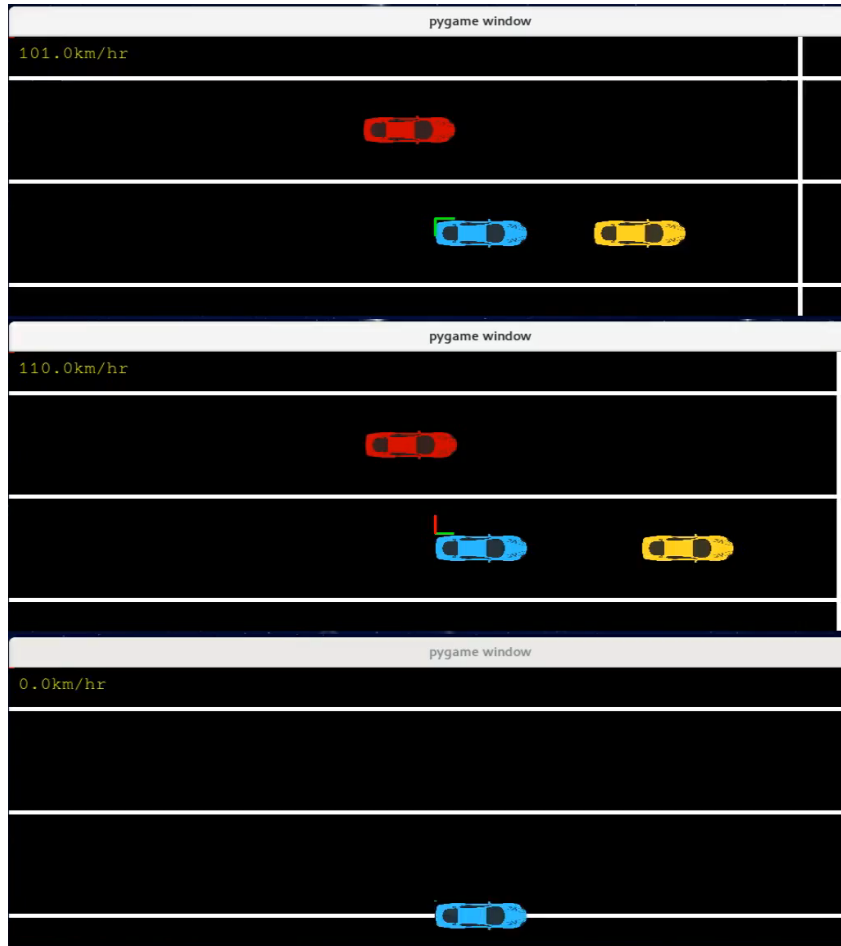
AALBORG UNIVERSITET



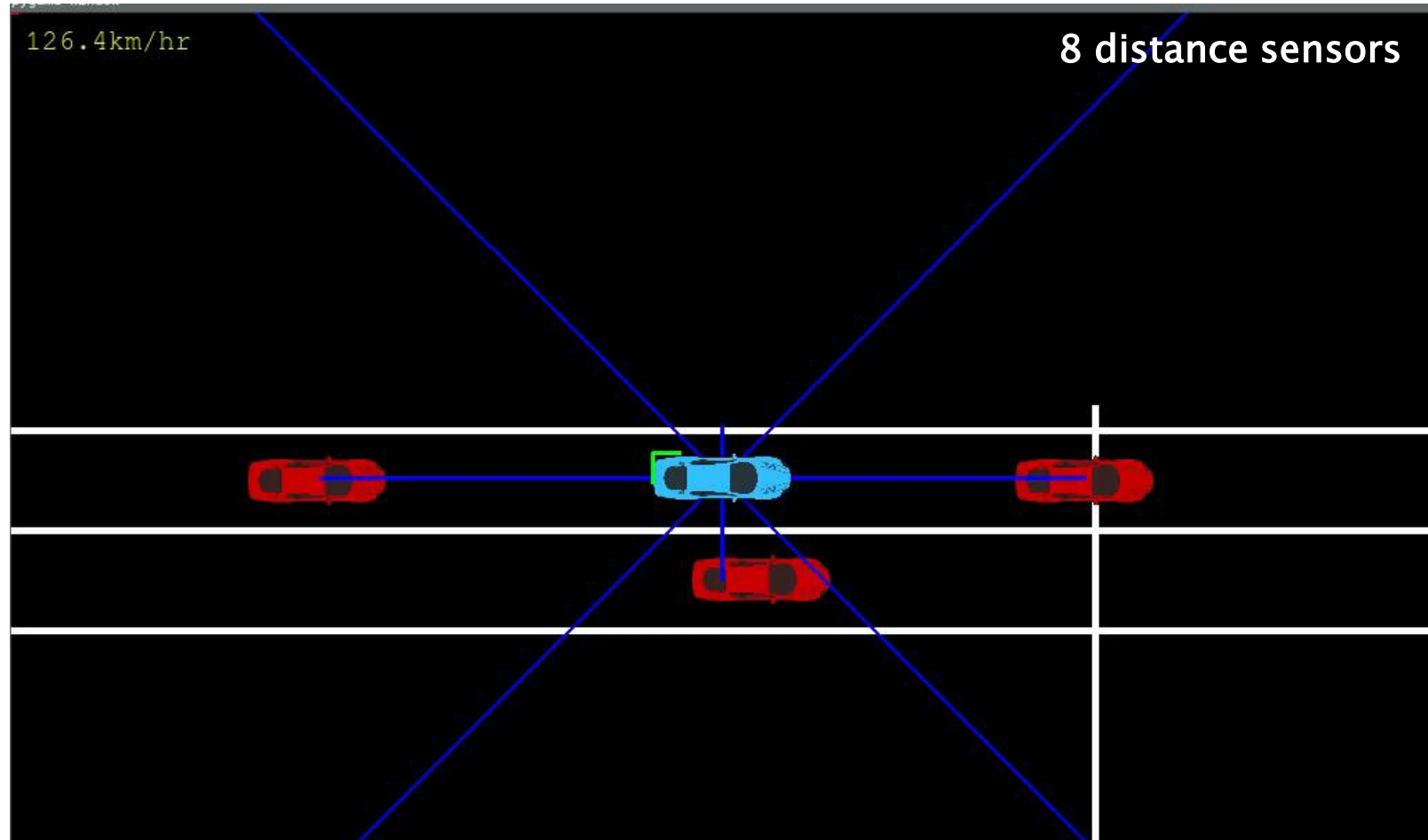
VILLUM FONDEN



# UPPAAL Adaptive Cruise Control

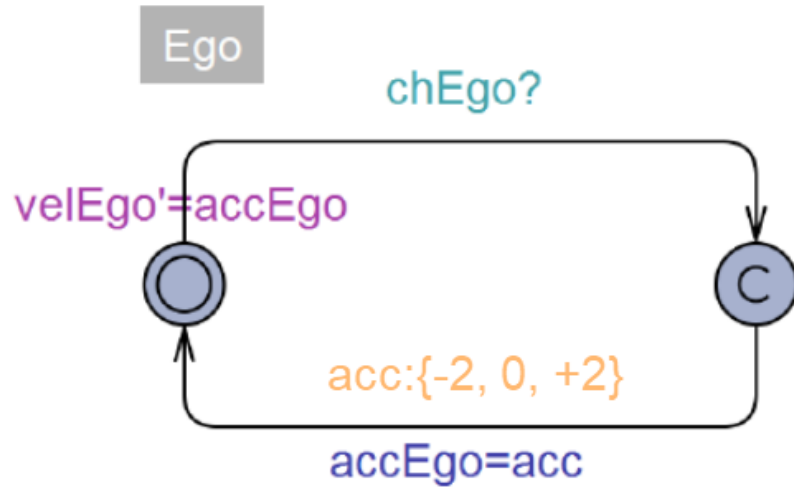


**“Optimal” Strategy**



**Safe and “Optimal” Strategy**

# Strategy – Explicit



```
adaptiveCruiseControl - Notepad
File Edit Format View Help

State: ( Ego.Negative_acc Front.No_acceleration System.Wait Monitor._id12 ) #action=0
distance=47 velocityEgo=6 accelerationEgo=-2 velocityFront=12 accelerationFront=0
While you are in      (waitTimer<=1), wait.

State: ( Ego.No_acc Front.Positive_acc System.Wait Monitor
velocityEgo=13 accelerationEgo=-2 velocityFront=14 accelerationFront=0
While you are in

State: ( Ego.No_acc Front.Positive_acc System.Wait Monitor
distance=199 velocityEgo=13 accelerationEgo=-2 velocityFront=14 accelerationFront=0
When you are in      transition Ego.Choose->Ego.Negative_acc { 1, tau, accelerationEgo := 0
}
When you are in      transition Ego.Choose->Ego.No_acc { velocityEgo <
maxVel, accelerationEgo := 2 }
When you are in      transition Ego.Choose->Ego.Positive_acc { velocityEgo >
minVel, accelerationEgo := 2 }

State: ( Ego.Negative_acc Front.Choose System.Done Monitor._id12 ) #action=0 distance=199
velocityEgo=13 accelerationEgo=-2 velocityFront=14 accelerationFront=0
While you are in      true, wait.

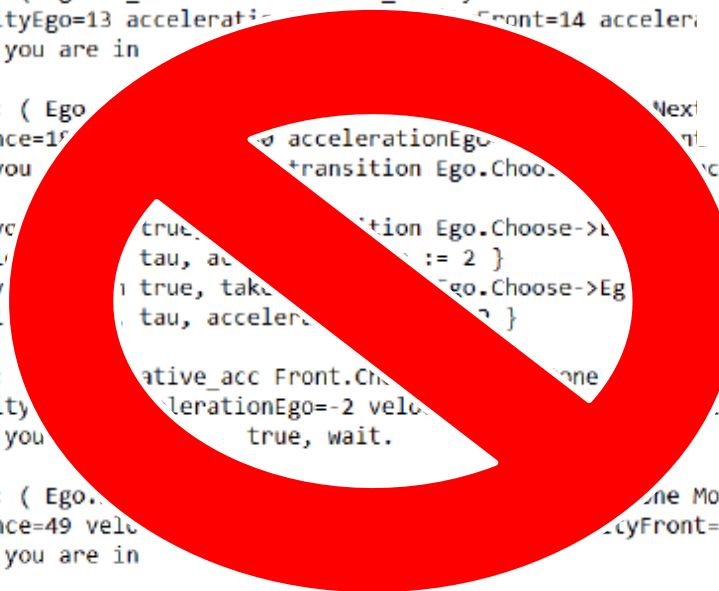
State: ( Ego.No_acc Front.Positive_acc System.Done Monitor._id12 ) #action=0
distance=49 velocityEgo=13 accelerationEgo=-2 velocityFront=14 accelerationFront=2
While you are in

State: ( Ego.Positive_acc Front.Choose System.Done Monitor._id12 ) #action=0 distance=88
velocityEgo=0 accelerationEgo=2 velocityFront=11 accelerationFront=0
While you are in      true, wait.

State: ( Ego.Positive_acc Front.Choose System.Done Monitor._id12 ) #action=0 distance=174
velocityEgo=18 accelerationEgo=2 velocityFront=17 accelerationFront=2
While you are in      true, wait.

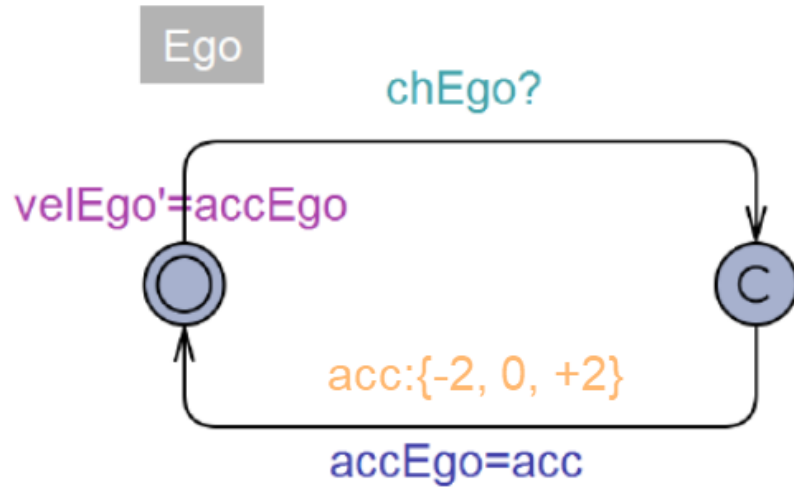
State: ( Ego.No_acc Front.Negative_acc System.Done Monitor._id12 ) #action=0 distance=147
```

4Mb  
6 mio configurations





# Strategy – Decision Tree



```

Ego.Choose <= 0: 3 (1481817.0)
Ego.Choose > 0
| velocityEgo <= -10: 0 (39705.0/18380.0)
| velocityEgo > -10
| | distance <= 200
| | | velocityEgo <= 18
| | | | velocityEgo <= 12
| | | | | distance <= 184
| | | | | velocityEgo <= 0: 2 (331464.0/20857
| | | | | velocityEgo > 0
| | | | | | distance <= 122
| | | | | | | distance <= 102: 2 (132918.0/80
| | | | | | | distance > 102
| | | | | | | | velocityEgo <= 2
| | | | | | | | | velocityFront <= 12: 1 (62500.0/1000.0)
| | | | | | | | | velocityFront > 12
| | | | | | | | | | velocityFront <= 13: 2 (162.0)
| | | | | | | | | | velocityFront > 13
| | | | | | | | | | | distance <= 110: 2 (870.0/363.0)
| | | | | | | | | | | distance > 110
| | | | | | | | | | | | velocityFront <= 15
| | | | | | | | | | | | | velocityFront <= 14: 1 (207.0/99.0)
| | | | | | | | | | | | | velocityFront > 14: 2 (63.0)
| | | | | | | | | | | | | velocityFront > 15
| | | | | | | | | | | | | | distance <= 116
| | | | | | | | | | | | | | | velocityFront <= 17: 2 (126.0/54.0)
| | | | | | | | | | | | | | | velocityFront > 17: 1 (129.0/39.0)
| | | | | | | | | | | | | | | | distance > 116: 1 (108.0)
| | | | | | | | | | | | | | | | | velocityEgo > 2
| | | | | | | | | | | | | | | | | | velocityEgo > 4: 1 (7680.0/1020.0)

```

## Learning Algorithms for Decision Tree (ID3, D4.5, CART)

→ 65 lines

Jan Kretinsky, Pranav Ashkot, TUM

[QEST19] SOS: Synthesis for Hybrid MDP

[TACAS21]

dtControl 2.0: Explainable Strategy Representation via Decision Tree Learning Steered by Experts.

Safe

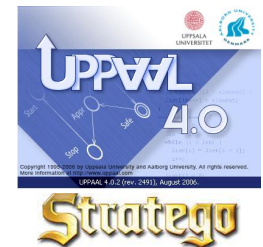
# Applications



AALBORG UNIVERSITET

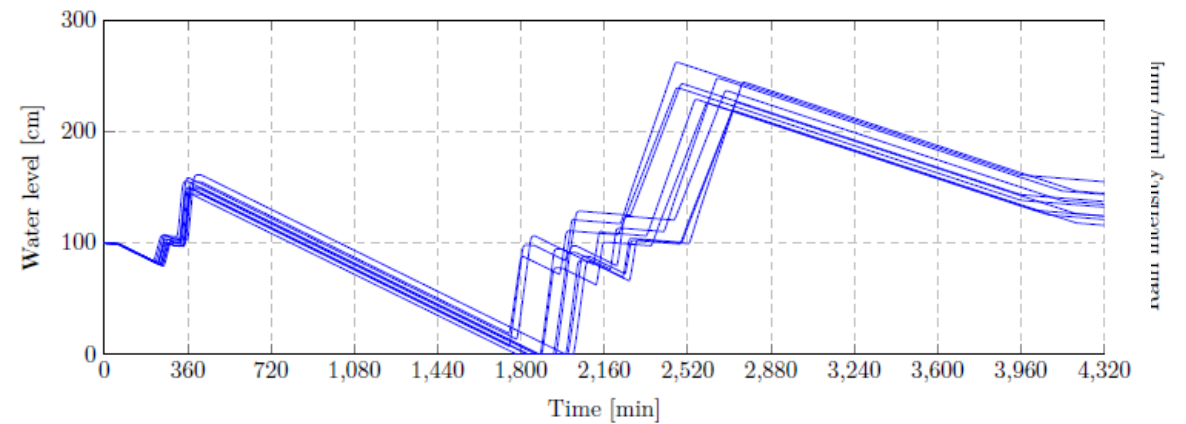
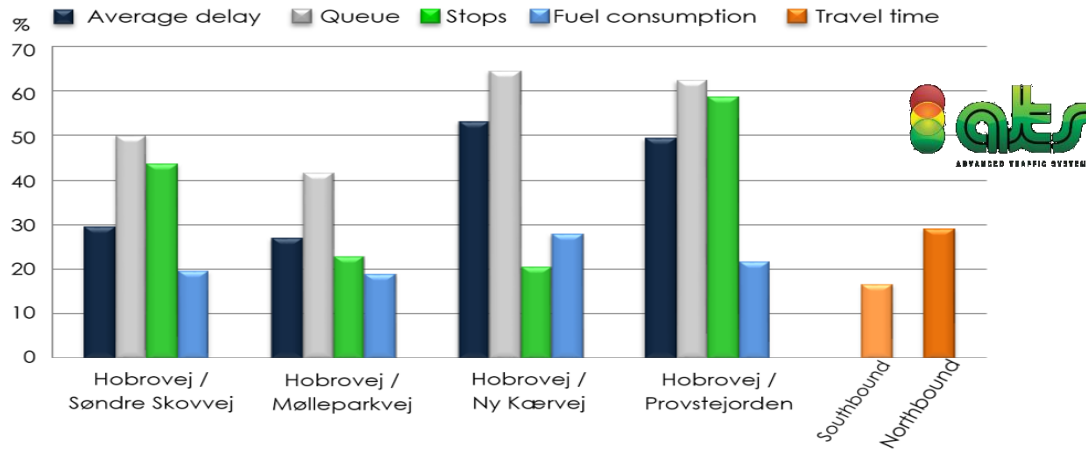
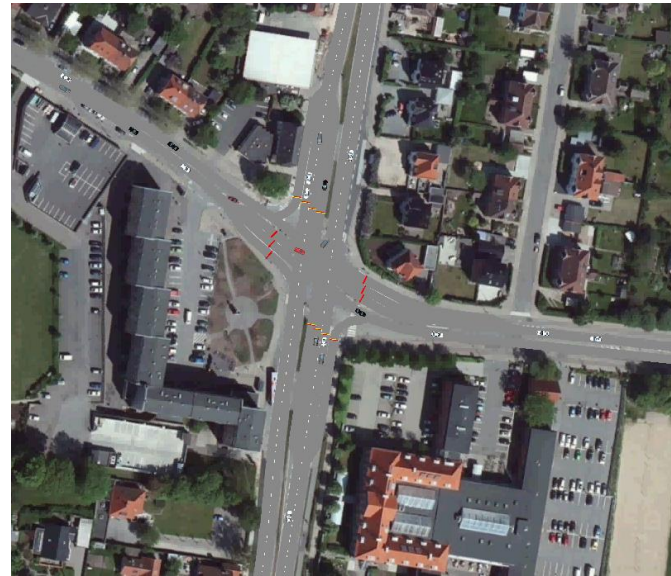


VILLUM FONDEN



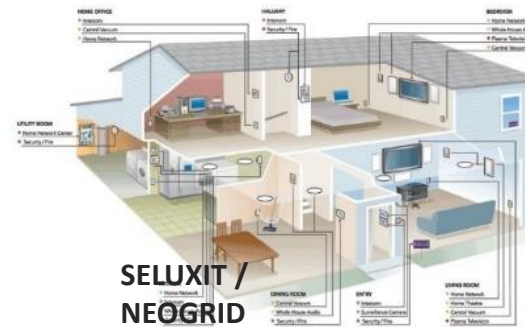


# Intelligent Transport -- Smart Water



# More Synthesis & Ongoing Research

- Convergence of Q-learning for IMDP.
- Partial Observability
- Learning strategy profiles for composite systems
- Online/Offline
- ..



VILLUM FONDEN

