

SON Conflict Resolution using Reinforcement Learning with State Aggregation

Ovidiu Iacobaiea^{†‡}, Berna Sayrac[†], Sana Ben Jemaa[†], Pascal Bianchi[‡]

([†])Orange Labs, 38-40 rue du General Leclerc 92130, Issy les Moulineaux, France

([‡]) Telecom ParisTech, 37 rue Dareau 75014, Paris, France

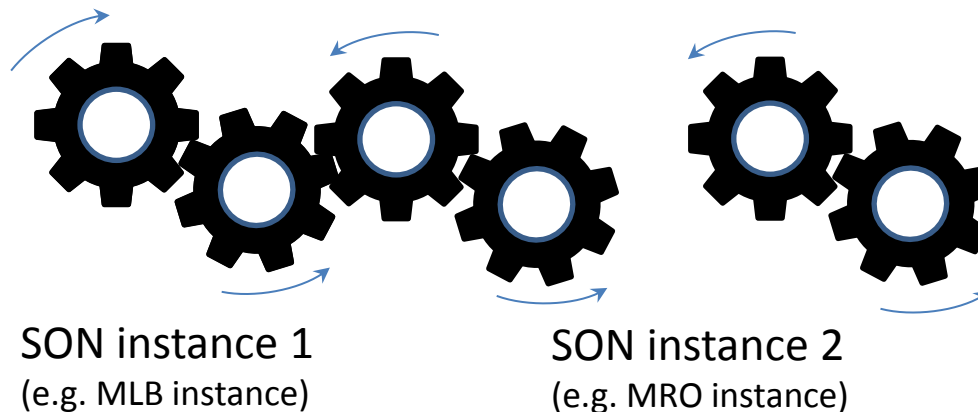


Presentation agenda:

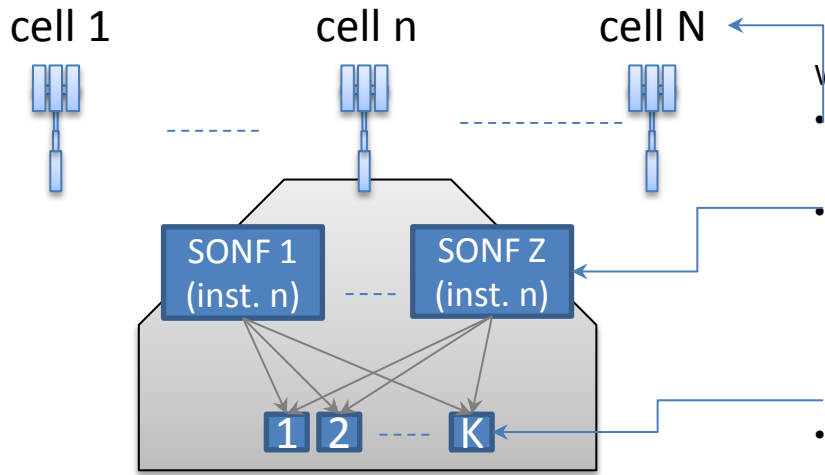
- Introduction
- System Description: SONCO, parameter conflicts
- Reinforcement Learning
- State Aggregation
- Simulation Results
- Conclusions and Future Work

Introduction to SON & SON Coordination

- ❑ Self Organizing Network (SON) functions are meant to automate network tuning (e.g. Mobility Load Balancing, Mobility Robustness Optimization, etc.) in order to reduce CAPEX and OPEX.
- ❑ A SON instance is a realization/instantiation of a SON function running on one (or several) cells.
- ❑ In a real network we may have several SON instances of the same or different SON functions, **this can generate conflicts.**
- ❑ Therefore we need a SON COordinator (SONCO)



System description



We consider:

- N cells. (each sector constitutes a cell)
- Z SON functions (e.g. MLB*, MRO*), black-boxes
 - each of which is instantiated on every cell, i.e. we have NZ SON instances
 - SON instances are considered as black-boxes
- K parameters on each cell tuned by the SON functions (e.g. CIO*, HandOver Hysteresis)

❑ The network at time t :

$P_{t,n,k}$ - the parameter k on cell n

❑ The SON at time t :

$U_{t,n,k,z} \in [-1; 1] \cup \{void\}$ - the request of (the instance of) SON function z targeting $P_{t,n,k}$

- $U_{t,n,k,z} \in [-1; 0)$, $U_{t,n,k,z} \in (0; 1]$ and $U_{t,n,k,z} = 0$ is a request to decrease, increase and maintain the value of the target parameter, respectively
- $|u|$ signifies the criticalness of the update, i.e. how unhappy the SON instance is with the current parameter configuration
- we consider that u may also be *void* for the case when a SON function is not tuning a certain parameter

❑ The SONCO at time t :

$A_{t,n,k} \in \{\pm 1, 0\}$ - the action of the SONCO

- if $A_{t,n,k} = 1 / A_{t,n,k} = -1$ means that we increase/decrease the value of $P_{t,n,k}$ only if there exists a SON update request to do so, else we maintain the value of $P_{t,n,k}$.

• targets to arbitrate conflicts caused by requests targeting the same parameters

(*) MLB = Mobility Load Balancing; (*) MRO = Mobility Robustness Optimization; (*) CIO = Cell Individual Offset

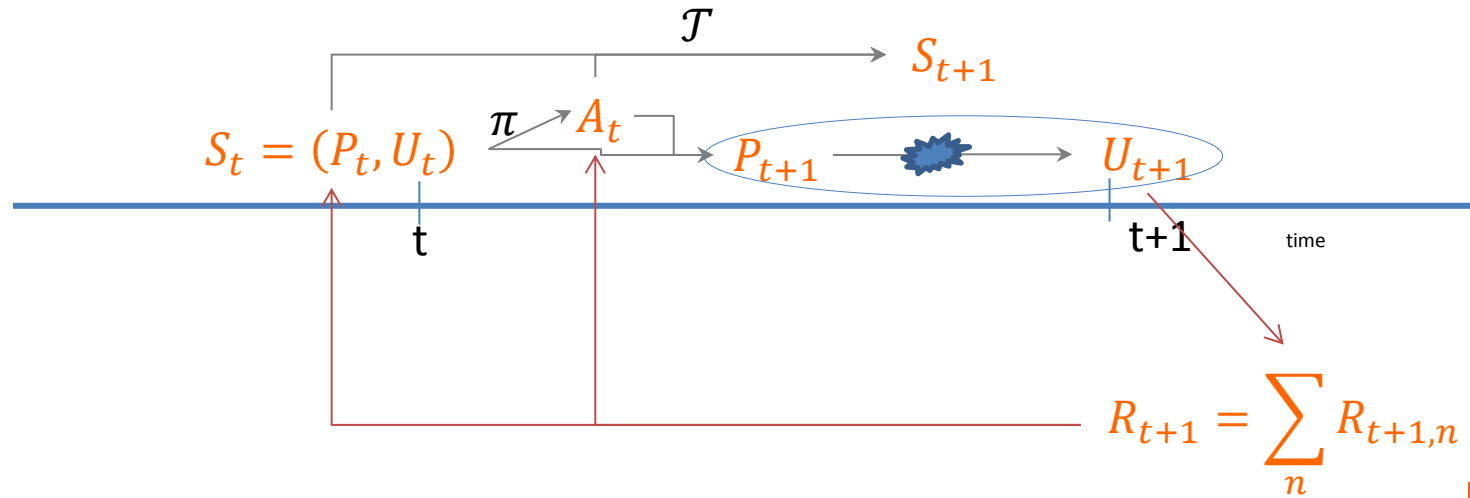
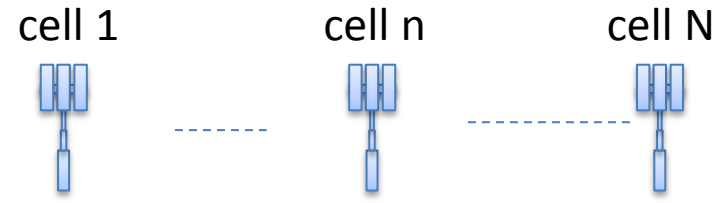
MDP formulation

□ State: $S_t = (P_t, U_t)$

□ Action: $A_t \in \{\pm 1, 0\}^{NK}$

□ Transition kernel:

- $P_{t+1} = g(P_t, U_t, A_t)$ (where g is a deterministic function)
- $U_{t+1} = h(P_{t+1}, \xi_{t+1})$, i.e. is a “random” function of P_{t+1} , and some noise ξ_{t+1}



e. g. $R_{t+1,n} = \max_{k,z} |U_{t+1,n,k,z}|$

Target: optimal policy, i.e. best A_t

- we define discounted sum regret (value function):

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_t \mid S_0 = s \right], 0 \leq \gamma \leq 1$$

- the optimal policy π^* is the policy which is better or equal to all other policies:

$$V^{\pi^*}(s) \leq V^\pi(s), \quad \forall s$$

- the optimal policy can be expressed as

$$\pi^*(s) = \underset{a}{\operatorname{argmin}} Q^*(s, a)$$

where $Q^*(s, a)$ is the optimal action-value function:

$$Q^*(s, a) = \mathbb{E}_{\pi^*} \left[\sum_{t=0}^{\infty} \gamma^t R_t \mid S_0 = s, A_0 = a \right]$$

- We only have partial knowledge of the transition kernel $\rightarrow Q^*$ cannot be calculated it has to be estimated (Reinforcement Learning). For example we could use Q-learning. BUT: we have deal with the complexity issue

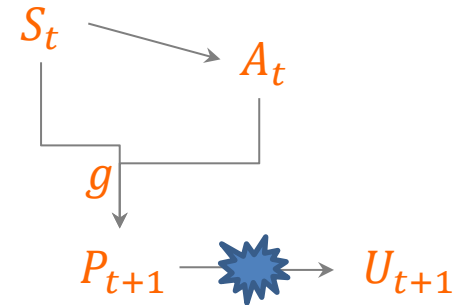
Towards a reduced complexity RL algorithm

Main idea : exploit the particular structure/features of the problem/model:

❑ Special structure of the transition kernel:

$$P_{t+1} = g(S_t, A_t)$$

$$U_{t+1} = h(P_{t+1}, \xi_{t+1})$$



❑ the regret:

$$R_{t+1} = \sum_{n \in \mathcal{N}} R_{t+1,n}$$

only depends on

The consequence is:

$$Q(s, a) = \sum_{n \in \mathcal{N}} W_n(p'), p' = g(s, a)$$

The complexity is reduced as now we can learn the W -function instead of the Q -function, (the domain of $(s, a) = ((p, u), a)$ is smaller than the domain of $g(s, a) = p$)

Still not enough, but...

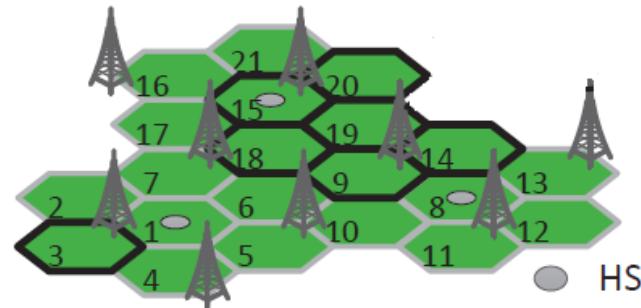
❑ The complexity is still too large as the domain of $\mathbf{p}' = g(s, a)$ scales exponentially with the number of cells.

➔ Use state aggregation to reduce complexity.

$$W_n(p) \approx \bar{W}_n(\bar{p}_n)$$

\bar{p}_n contains the parameters of cell n and its neighbors, which are the main cause of conflict.

e.g. in our example: keep the CIO and eliminate the Handover Hysteresis.

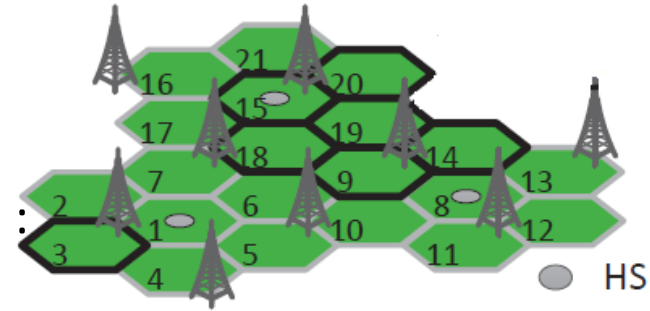


Application example

Some scenario details:

□ 2 SON functions instantiated on each and every cell :

- **MLB ($z = 1$)**: tuning the CIO ($k = 1$)
- **MRO ($z = 2$)**: tuning the CIO ($k = 1$) and the HandOver Hysteresis ($k = 2$)



□ we have a **parameter conflict** on the CIO

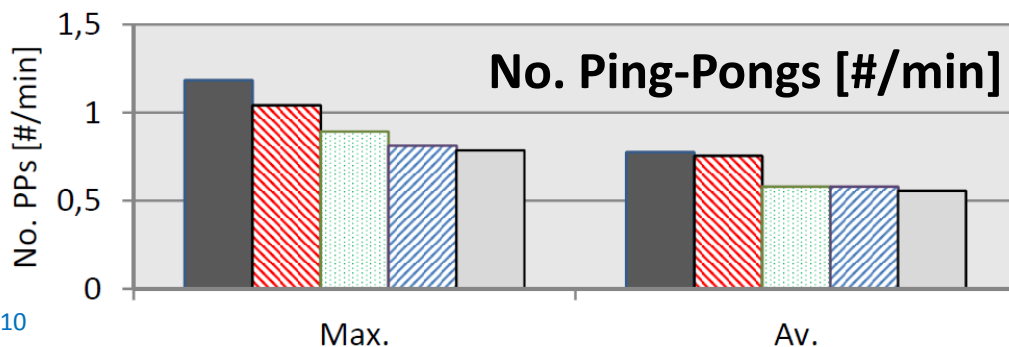
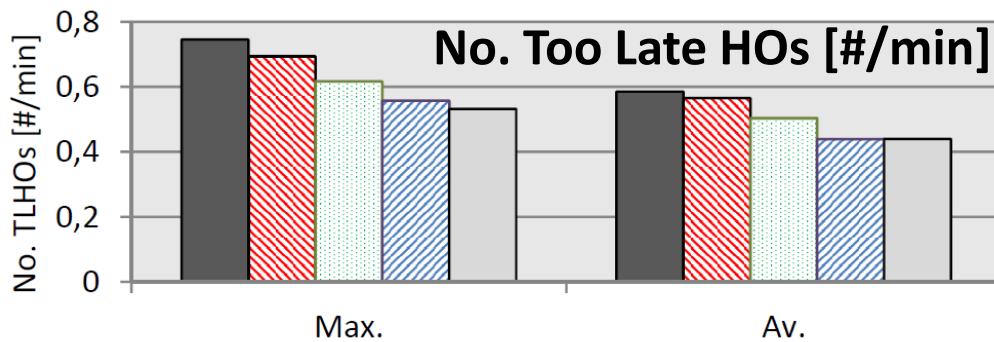
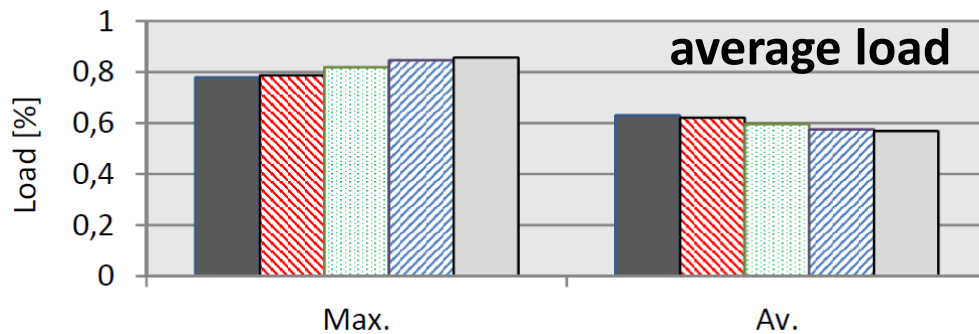
□ the regret is a sum of sub-regrets calculated per cell $R_{t,n} = \max_{k,z} |U_{t,n,k,z}| \rightarrow W_n (n \in \mathcal{N})$

□ from $W_n(p)$ to $\bar{W}_n(\bar{p}_n)$: \bar{p}_n contains the CIOs of cell n and its neighbors

□ consequence: the state space scales **linearly** with the no. of cells.

□ to be able to favor the SON functions in calculating the regret we also associate some **weights** to the SON functions

Simulation Results



MLB weight

MRO weight

High priority to MLB

High priority to MRO

- $w=(8,1)$
- $w=(4,1)$
- $w=(1,1)$
- $w=(1,4)$
- $w=(1,8)$

- $w=(8,1)$
- $w=(4,1)$
- $w=(1,1)$
- $w=(1,4)$
- $w=(1,8)$

- we have 48h of simulations
- the results are evaluated over the last 24h, when the CIOs become reasonably stable

Conclusion and future work

- ❑ we are capable of **arbitrating** in favor of one or another SON function (according to the weights)
- ❑ the solutions state space **scales linearly** with the number of cells
- ❑ still there remains a problem on the **action selection** (in the algorithm we exhaustively evaluate any possible action to find the best one)

Future work:

- analyzing **tracking** capability of the algorithm,
- **HetNet scenarios**,

Questions ?



ovidiu.iacobaiea@orange.com