Design and evaluation of self-optimisation algorithms for radio access networks

Orange Labs

Zwi Altman,
June 9th 2009, Santander Spain

orange™

Outline

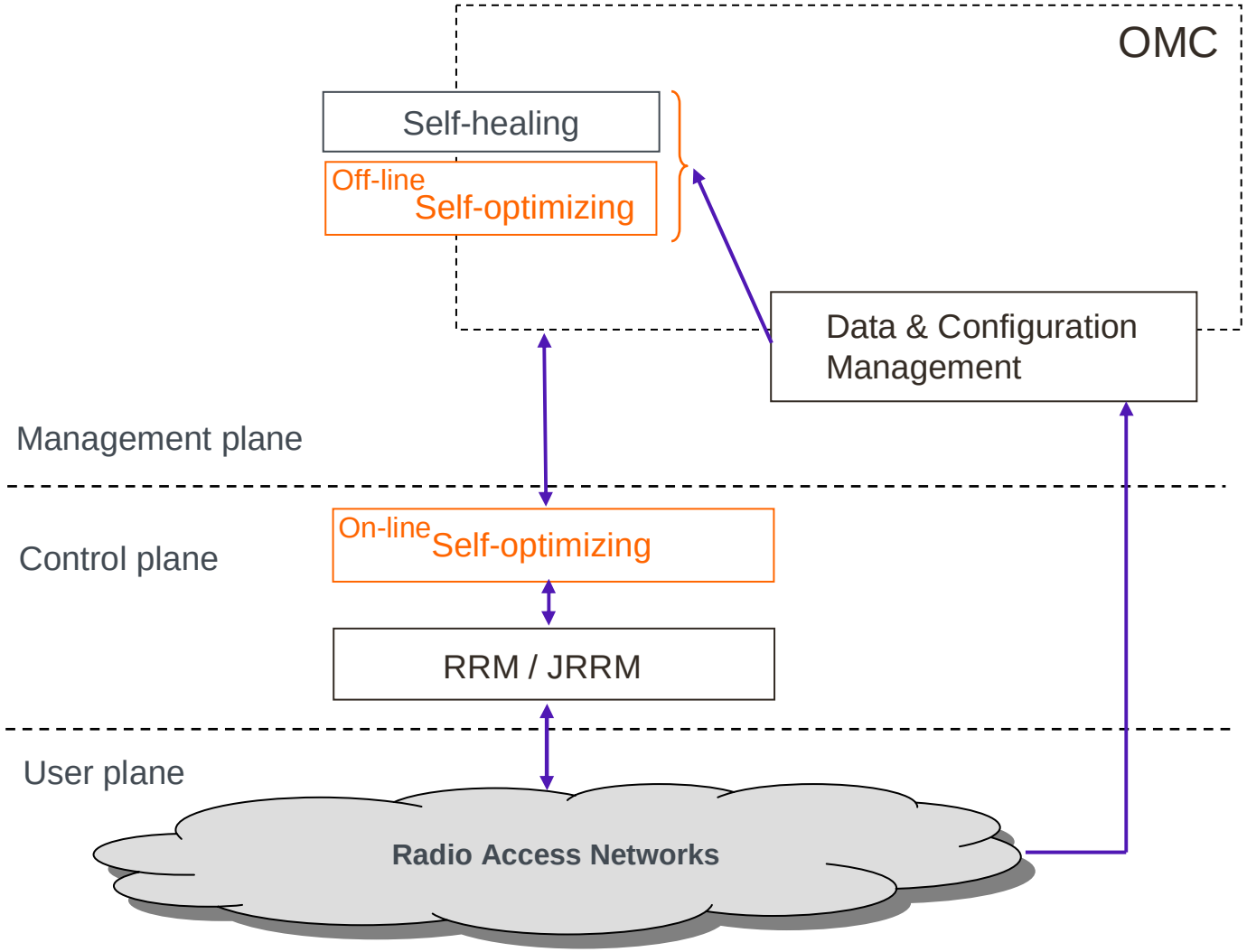
- Introduction:

  Control and optimization
- Reinforcement Learning framework
- Case study – LTE ICIC
- Conclusions

Self-Optimising Networks: Time scales

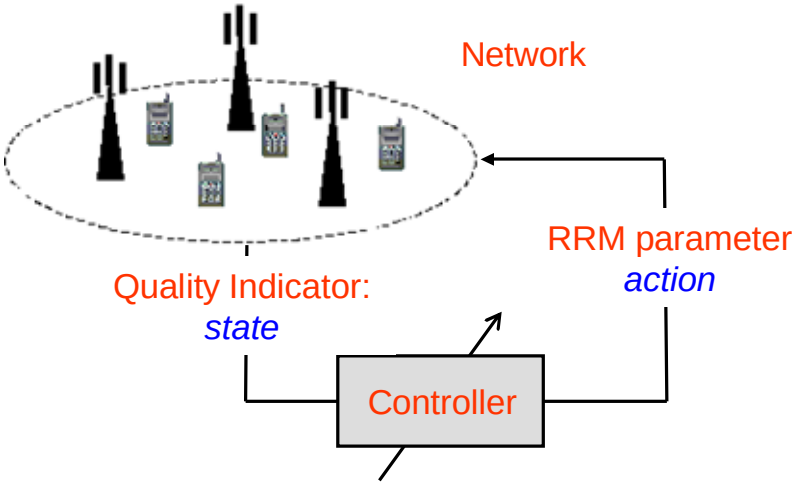
- Off-line process
  - Long time scale: hours, days
  - Optimization process (fixe RRM parameters for long period)
    - Statistical learning and optimization heuristics
- On-line process
  - Short time scale: seconds / minutes
    - Not too short to guarantee stability and
    - Based on trends and not on fluctuations
  - Dynamic control process

On-line and off-line Self-Optimisation



Control

- System state S
  - $S=(KPI_1,\ldots, KPI_N)$
  - KPI could be
    - load, number or users
    - throughput, spectral efficiency
    - QoS indicators (DCR, BCR, …)



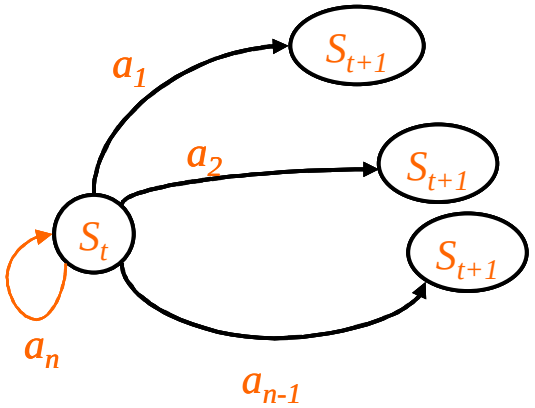
- Action *a*
  - Parameter change
    - Power (Δ) value
    - Threshold (Δ) value
    - Decision (access, handover)

- Control
  - Maps a sate into action

$$\pi(s): S \rightarrow A$$

… Control

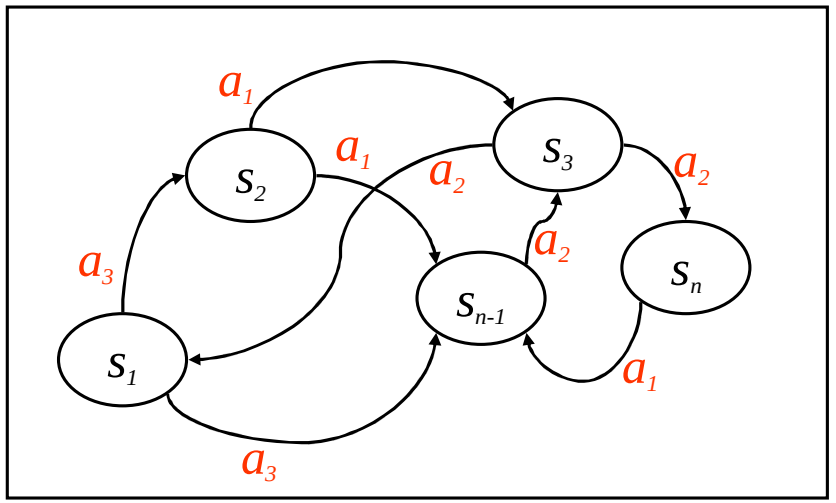
- Objective
  - At each state of the system, find best action among a given set of authorized actions
- What is best action ?
  - We seek to optimize $\pi(s)$, namely the full control process
- Policy
  - The control function $\pi(s)$ defined over all states



$$\pi(s): S \rightarrow A$$

… Control

- States / KPIs are noisy
  - Traffic and radio channel are of stochastic nature
  - Measurements have limited precision

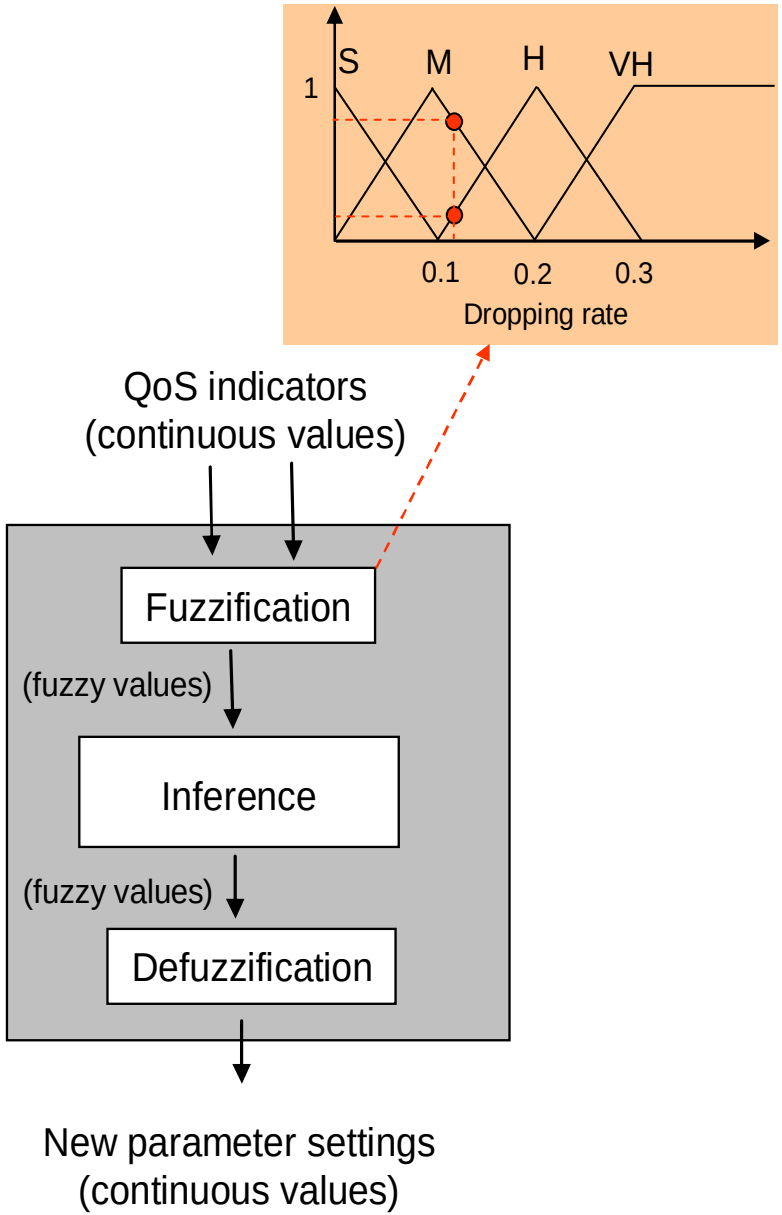


… Control for continuous state space

State space is often continuous

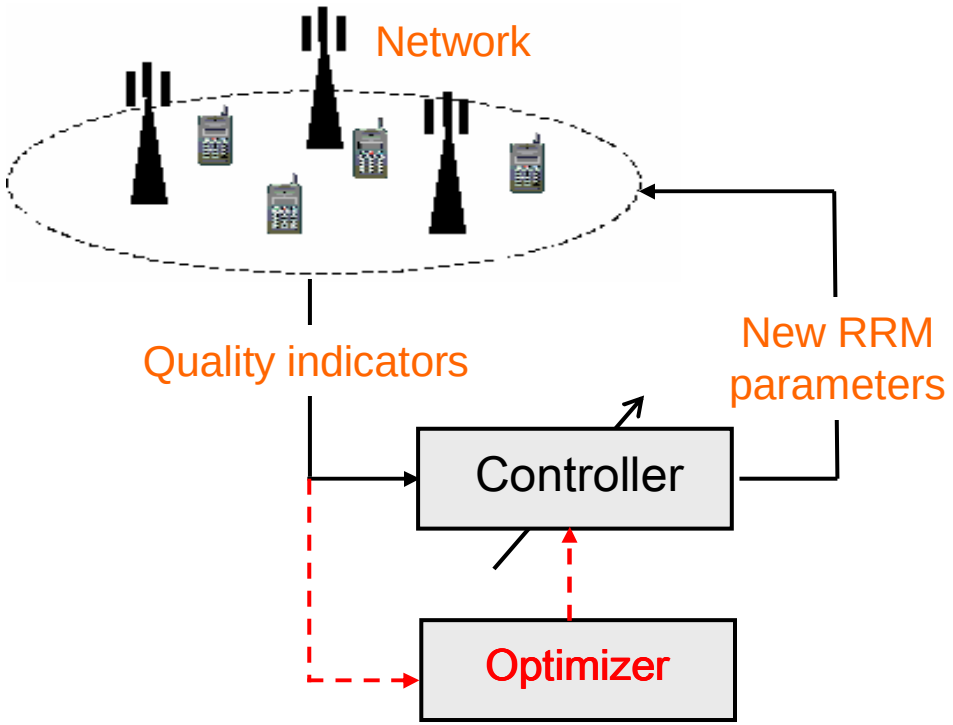
- Fuzzy Logic Control (FLC) is often used for control tasks with continuous state space
  - It translates expert knowledge written in the form of if-then linguistic rules into mathematical form

IF (*DCR is high*) AND (*BCR is low*) THEN (*Big decrease of Thresh*$_{AC}$)



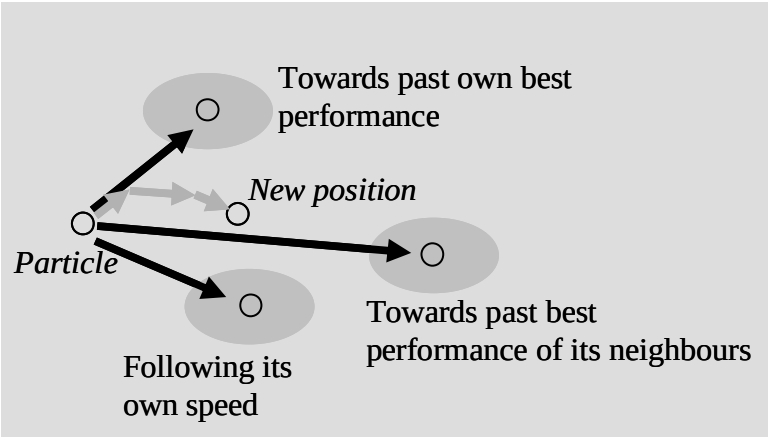
Optimizing the control process

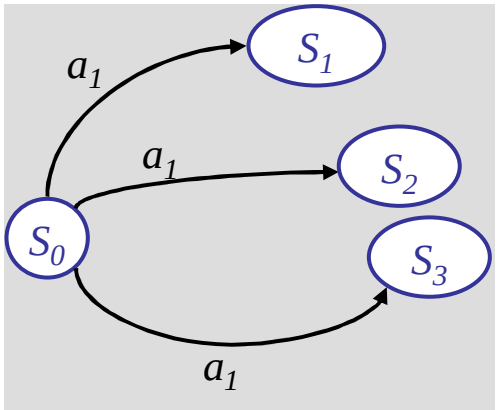
- Designing a controller is complex
- Automatic optimization is needed



Solving optimal control

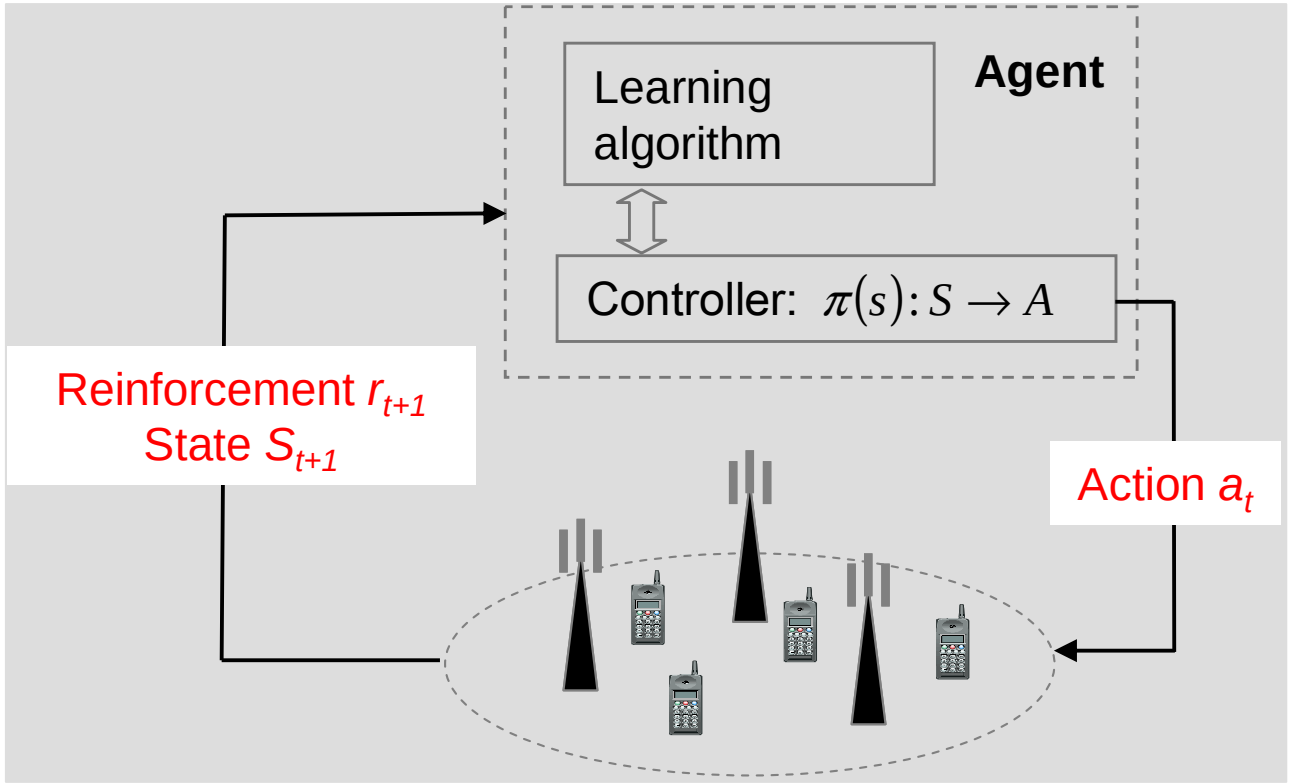
- Optimization heuristics
  - Particle swarm
  - Genetic algorithms
  - Simulated annealing
  - …
- Difficulty
  - Make the heuristic work with noisy data ?
  - Real implementation ?





Z. Altman at al, "Auto--tuning of RRM parameters in UMTS networks", chapter 16 in book *Understanding UMTS Radio network modelling, planning and automated optimisation: theory and practice*, Editors: M. Nawrocki, M. Dohler and H. Aghvami, Wiley, 2006.

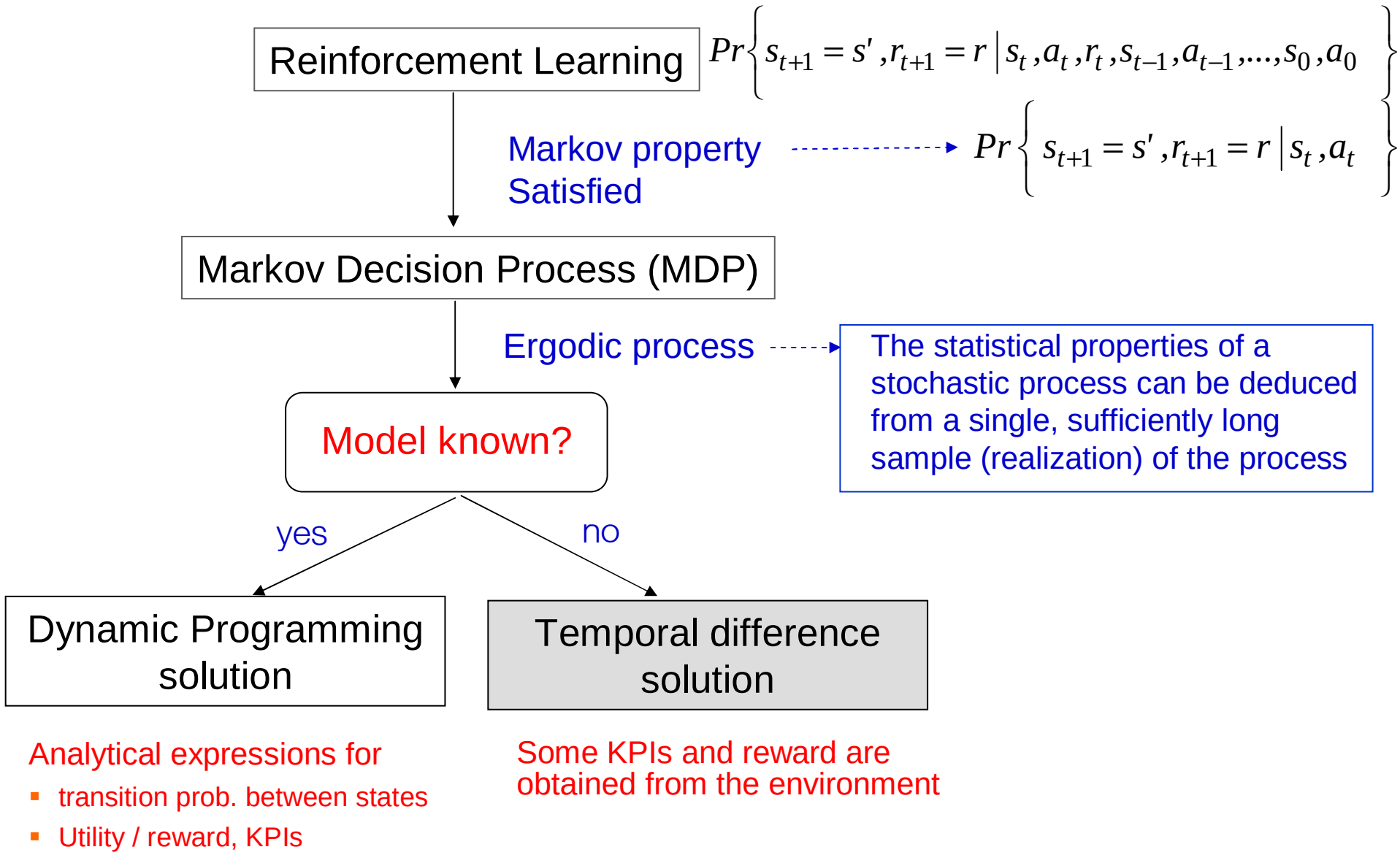
Reinforcement Learning (RL) framework for Self-Optimization



- Why RL ?
  - Robust to noise
  - Simple (but needs good understanding!)

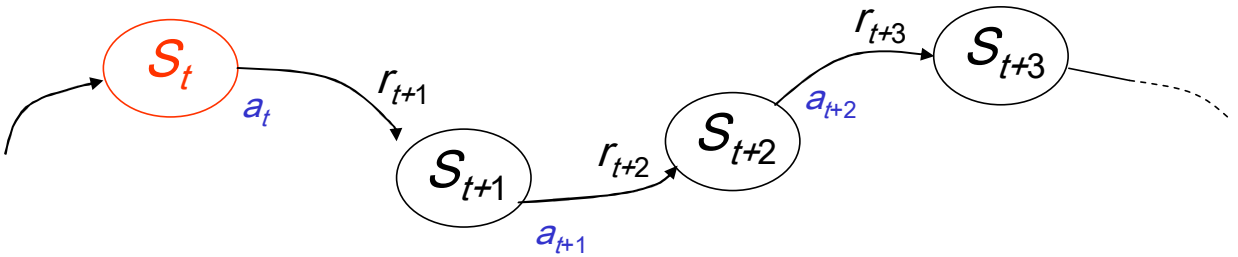
R.S. Sutton, A.G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998.

Reinforcement Learning approaches



Learning objective

- Finds a policy $\pi(s): S \rightarrow A$ that maximizes the sum of future cumulative rewards



- Without a model (temporal difference)

$$\max_{\pi \in \Pi} : R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad ; \quad 0 < \gamma < 1$$

- With a model

$$\max_{\pi \in \Pi} : R_t = E\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \right]$$

Q-algorithm

- Q-function

$$Q_t(s_t, a_t) = R_t(s_t, a_t) = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$ starting from (s_t, a_t)

- We choose an action a_t, and continue with the (up to now) best policy

$$Q_{t+1}(s_t, a_t) = r_{t+1} + \gamma \max_{a'} Q_t(s_{t+1}, a')$$

- To guarantee convergence: add a learning rate coefficient η

$$Q_{t+1}(s_t, a_t) = (1-\eta) Q_t(s_t, a_t) + \eta \left(r_{t+1} + \gamma \max_{a'} Q_t(s_{t+1}, a') \right)$$

$$= Q_t(s_t, a_t) + \eta \left(r_{t+1} + \gamma \max_{a'} Q_t(s_{t+1}, a') - Q_t(s_t, a_t) \right)$$

$$= Q_t(s_t, a_t) + \Delta Q_{t+1}$$

Exploration – exploitation policy

EEP – exploration exploitation policy

- Exploration: "*e-greedy policy*"

$$a_t = \begin{cases} \underset{a \in A}{argmax}\, Q_t(s_t, a) & ; \quad \text{with probability } 1-\varepsilon \\ \underset{a \in A}{rand}(a) & ; \quad \text{with probability } \varepsilon \end{cases}$$

- Exploitation (after learning is over)
  - The optimal policy is performed choosing the action that, at every state, maximizes the Q-function:

$$a = \underset{b \in A}{arg\ max}\, Q^*(s,b)$$

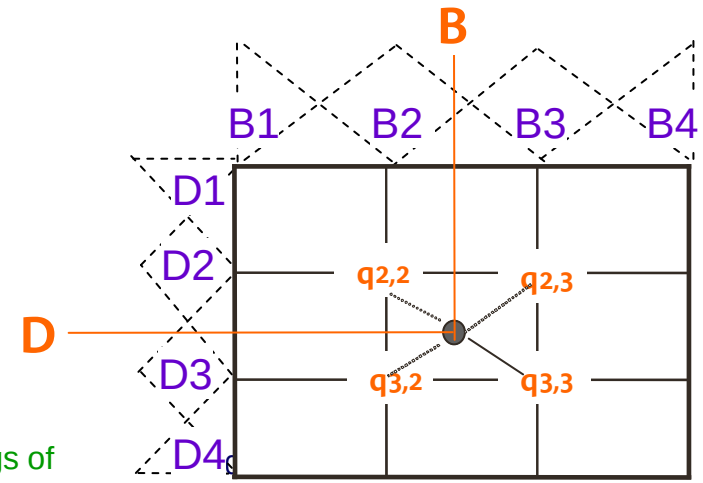
Q-learning algorithm

1. Set t=0 and initialize $Q_0(s_t, a) = 0$ for all $s_t \in S, a_t \in A$
2. Choose initial state s_t
3. Repeat until convergence
4. $a \leftarrow EEP(s_t, Q_t, S, A)$
5. Perform action a, leading to a new state s_{t+1}, with a reinforcement r_{t+1}
6. $$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \eta \left\{ r_{t+1} + \gamma \max_{a \in A} Q_t(s_{t+1}, a) - Q_t(s_t, a_t) \right\}$$
7. Set $s_t = s_{t+1}, t = t + 1$
8. End repeat

Fuzzy Q-Learning (FQL)

- In most problems, the state space is continuous
- FQL introduces interpolation in the QL using fuzzy logic:

1. The states are described using fuzzy sets
2. q functions are defined on a set of discrete points: the modal points of the fuzzy sets that define the rules
3. At each iteration the q functions over the discrete states are updated
4. Q(*s*,*a*) is calculated by interpolating the *q*-values (i.e. the closest ones, those with non-zero degree of membership of *s*).



P.Y. Glorennec, "Reinforcement learning: an overview", Proceedings of the ESIT 2000 conference, Aachen, Germany, Sept. 2000.

… Fuzzy Q-learning

- The action and the quality of an input state *s* and action *a* can be expressed as an interpolation of the quality in each rule

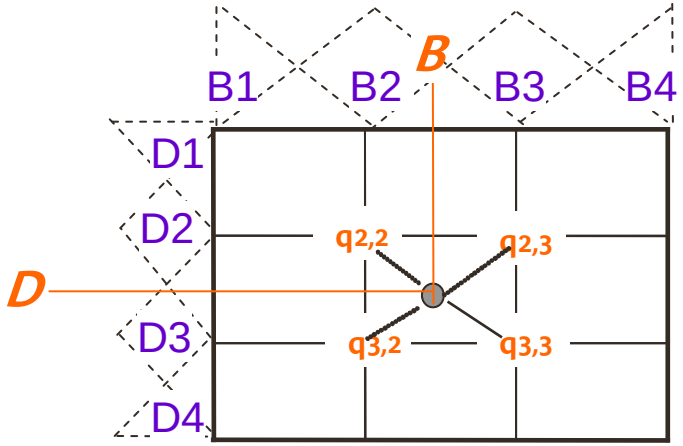
$$Q(s,a)=\sum_{j=1}^{m}\alpha_j(s)\times q_j\left(j,o_j\right)$$

$$\alpha_j(s)=\prod_{i=1}^{n_j}\mu_{ij}(s_i)$$

m-# excited rules
n_j- dimension of rule *j*

- $\alpha_j(s)$ is the distance (degree of truth) between the input state *s* and the rule *j*.

S=(*B*,*D*)



… Fuzzy Q-learning algorithm

$$(1)\ Q(s,a)=\sum_{j=1}^{m}\alpha_j(s)\times q_j\left(j,o_j\right)$$

$$(2)\ \alpha_j(s)=\prod_{i=1}^{n_j}\mu_{ij}(s_i)$$

m-# excited rules

n_j- dimension of rule *j*

Initialize a *q*-look-up table: $q(j,o_k)=0$; $1\le j\le R$ and $1\le k\le K_j$

Choose initial state *s*(0)

Repeat until convergence

Use EEP policy to determine the action o_j

with a probability 1-**ε**, for each activated rule, select an action o_j

$$o_j=\underset{k\in\{1,\ldots,K_j\}}{arg\,max}\,q(j,o_k)$$

with a probability **ε**, for each activated rule, select an action o_j

$$o_j=\underset{k\in\{1,\ldots,K_j\}}{rand}\{o_k\}$$

Choose an action

Execute action *a*, $a=\sum_{j=1}^{m}\alpha_j(s)\times o_j$ and observe

Observe new state *s*(*t*+1) and reinforcement signal *r*(*t*+1)

Compute $\alpha_j(s(t+1))$ for all rules R_j

Calculate the value of the new state: $V_t(s(t+1))=\sum_{j=1}^{m}\alpha_j(s(t+1))\cdot\max_{k}q(j,o_k)$

Calculate the variation of the quality $Q(s,a)$: $\Delta Q=r(t+1)+\gamma\times V_t(s(t+1))-Q(s(t),a)$

with *Q*(*s*(*t*),*a*) given by (1)

Update $q(j,o_j)$ for each rule *j* and action o_j: $q(j,o_j)=q(j,o_j)+\eta\cdot\Delta Q\cdot\alpha_j(s(t))$.

Update equation

Set *s*(*t*)=*s*(*t*+1)

End repeat

Convergence

$$Q_{t+1}(s_t, a_t) = (1-\eta_t) Q_t(s_t, a_t) + \eta_t \left\{ r_{t+1} + \gamma \max_{a \in A} Q_t(s_{t+1}, a) \right\}$$

- Q-Learning guarantee that the Q values converge to those of the optimal policy if
  - each state-action pair is sampled on infinite number of times,
  - the learning rate η_t verifies

$$0 \leq \eta_t < 1$$

$$\sum_{0}^{\infty} \eta_t = \infty$$

$$\sum_{0}^{\infty} \eta_t^2 < \infty$$

Stability

- To improve stability, all quantities (states and rewards) are filtered using an averaging sliding widow
  - The auto-tuning process should rely on long term tendencies, not on short term fluctuations

$$I_{filtered}(t) = \frac{1}{T}\sum_{n=0}^{T-1} I(t-n)$$

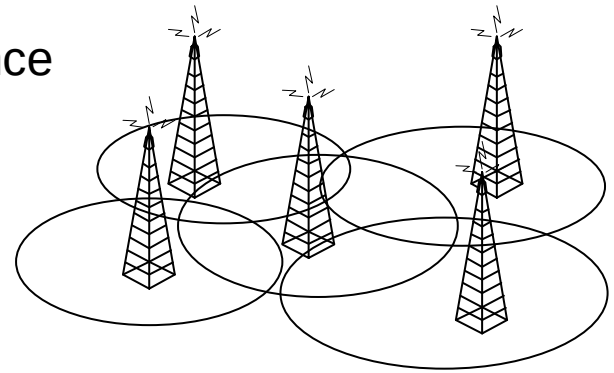
Rigorous MDP solution

- Hypothesis
  - All base stations in the network implement auto-tuning
- The rigorous solution with guarantee of convergence
  - *N* base stations implementing control

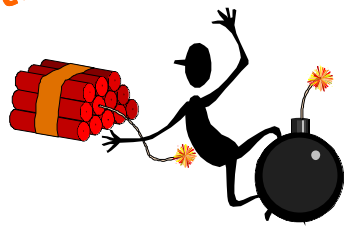
$$\vec{S} = (S_1, \cdots, S_N) \quad ; \quad \vec{a} = (a_1, \cdots a_N)$$

$$Q = Q(\vec{S}, \vec{a})$$

  - System with full information sharing
  - Simultaneous (synchronous) actions



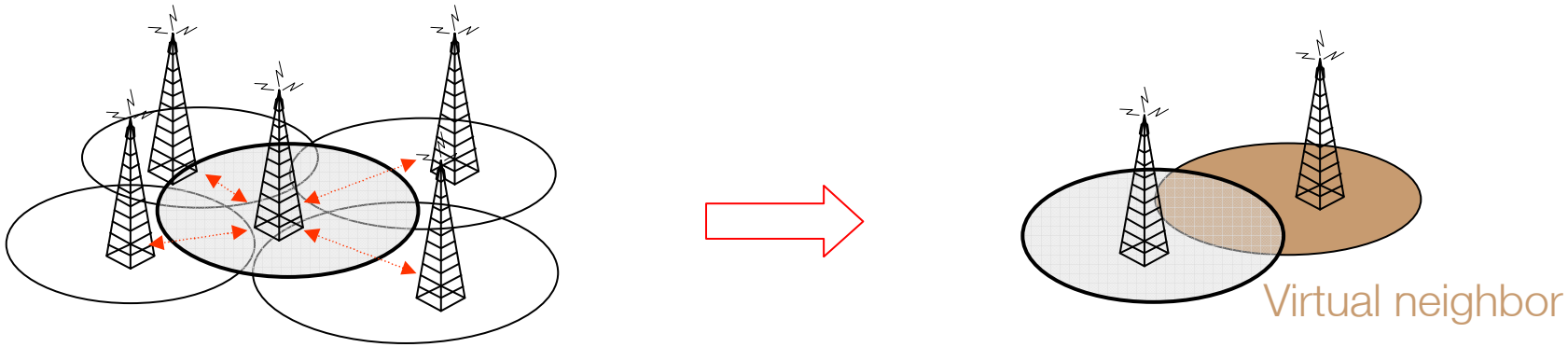
Not practical !



Approximated MDP: scalable solution

- Controlled sub-system: a base station and its neighbors
  - State $\vec{S} = \left(\vec{S}_{bs}, \vec{S}_{nbs}\right)$
  - Local action $a = a_{bs}$
  - Reinforcement: global or local

- Virtual neighbor



- A base station does not take into account the actions of its neighbors in the control
  - Approximated MDP
  - Scalable solution
  - Non-synchronous control

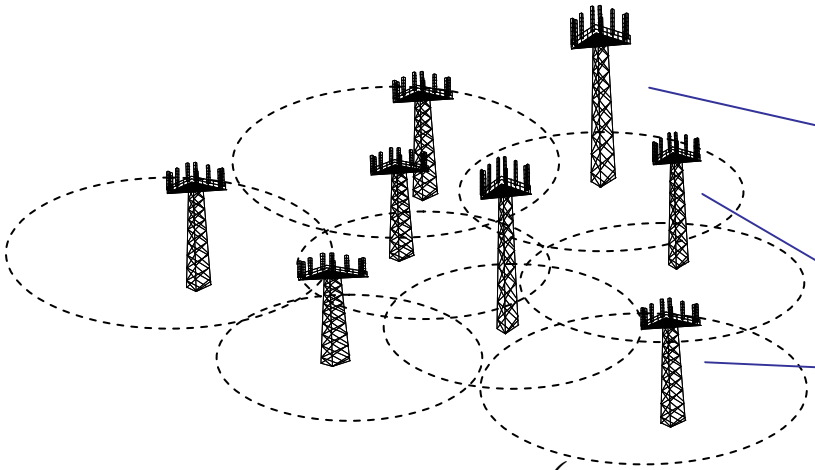
$$S_{nbs} = \sum_{i \in N(bs)} \omega_{bs,i} S_i$$

ω – normalized traffic flux

$N(bs)$ – neighbors of bs

Cooperative learning: implicit parallelism

- Cooperative learning (exploration phase)
  - Share experience => Feed the same *Q* table
  - Global or local reward
- Distributed exploitation
  - Each base station performs its own control



s	a	$Q(s, a)$

$$Q_{t+1}(s_t, a_t) = (1-\eta) Q_t(s_t, a_t) + \eta \left(r_{t+1} + \gamma \max_{a'} Q_t(s_{t+1}, a') \right)$$

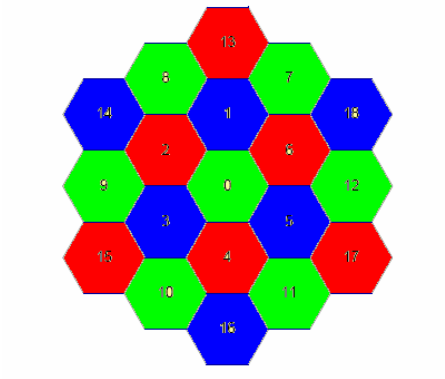
$$r_{t+1}(s_t^i, a_t) \qquad r_{t+1} = \sum_{i=1}^{N_{stations}} r_{t+1}^i(s_t^i, a_t)$$

Use case

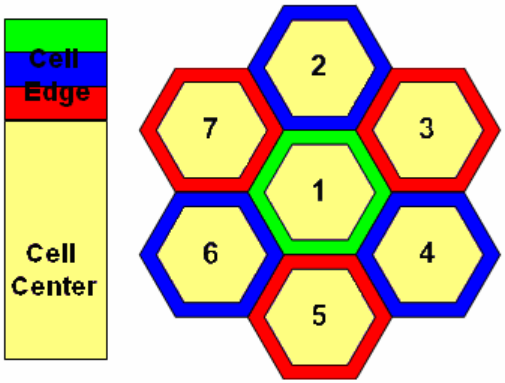
- SON use cases studied
  - Admission control (UMTS)
  - Mobility (UMTS)
  - RT and NRT resource allocation (UMTS)
  - Mobility in heterogeneous network (UMTS-WLAN)
  - Mobility (LTE)
  - DL ICIC (LTE)
  - UL Power control / ICIC (LTE)

Self-optimizing of ICIC in LTE network (downlink)

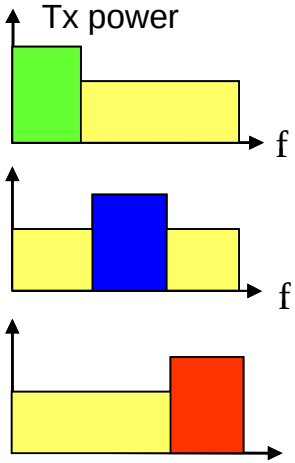
- Motivation
  - Different schemes have been developed to combat interference in OFDMA systems to mitigate interference (particularly in cell edge)



Reuse schemes (3)



Fractional reuse schemes

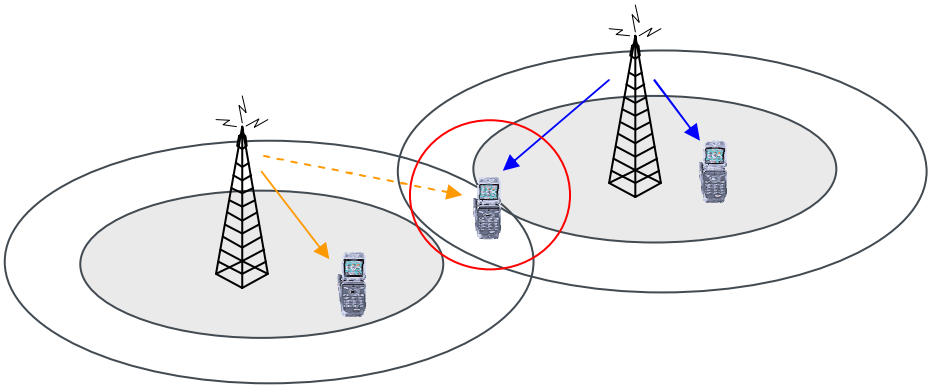


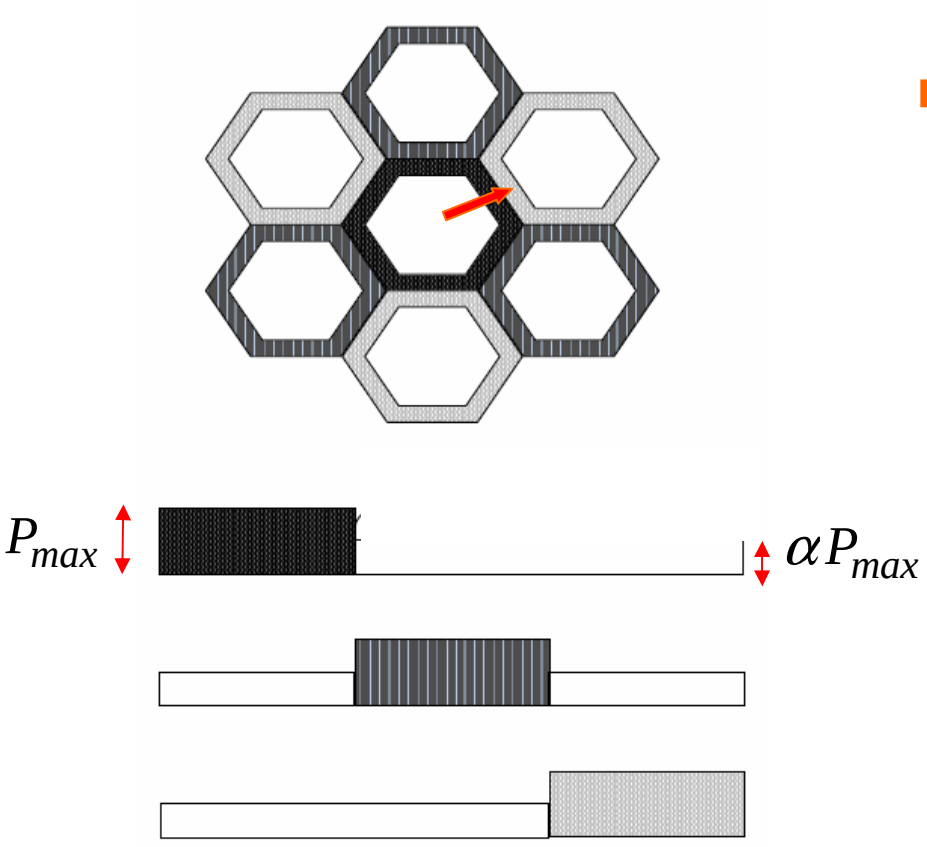
Soft reuse scheme

- Soft reuse scheme
  - Different power allocation is associated to different portion of the frequency bandwidth. The power allocation pattern is denoted as the power mask

Dynamic soft reuse scheme

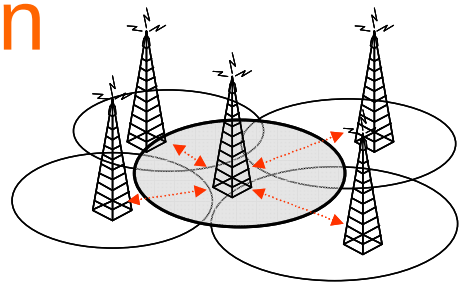
- Tranmission power is dynamically addapted to alleviate intererence of cell edge users





- Main interference
  - From "cell center users"
  - To "neighboring edge users"

Fuzzy Q-Learning solution



System states

$$S = \left(P_c^{bs}, SE_c^{bs}, SE_e^{nbs} \right)$$

averaged over 40 sec

$SE_{c/e}^{j}$ - Mean spectral efficiency of users of center / edge of cell *j*

$\omega_{bs,i}$ - normalized traffic flux between base station *bs* and *i*

$$I_{nbs} = \sum_{i \in N(bs)} \omega_{bs,i} I_i \qquad \sum_{i \in N(bs)} \omega_{bs,i} = 1$$

Action a

- *Power reduction* α

  Periodicity: 40 sec

$$P_c^{bs} = \alpha P_{max}$$

Reward

- *Harmonic throughput*
  - Optimize harmonic mean fairness
  - Associated to file transfer time

$$r = \sum_{k=1}^{N_{stations}} \sum_{u \in U(k)} \frac{1}{Th_{u,k}}$$

Solution evaluation

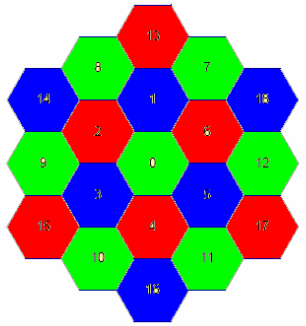
- Learning is performed for a given traffic intensity (arrival rate)
- Evaluation is performed over a range of traffic intensities

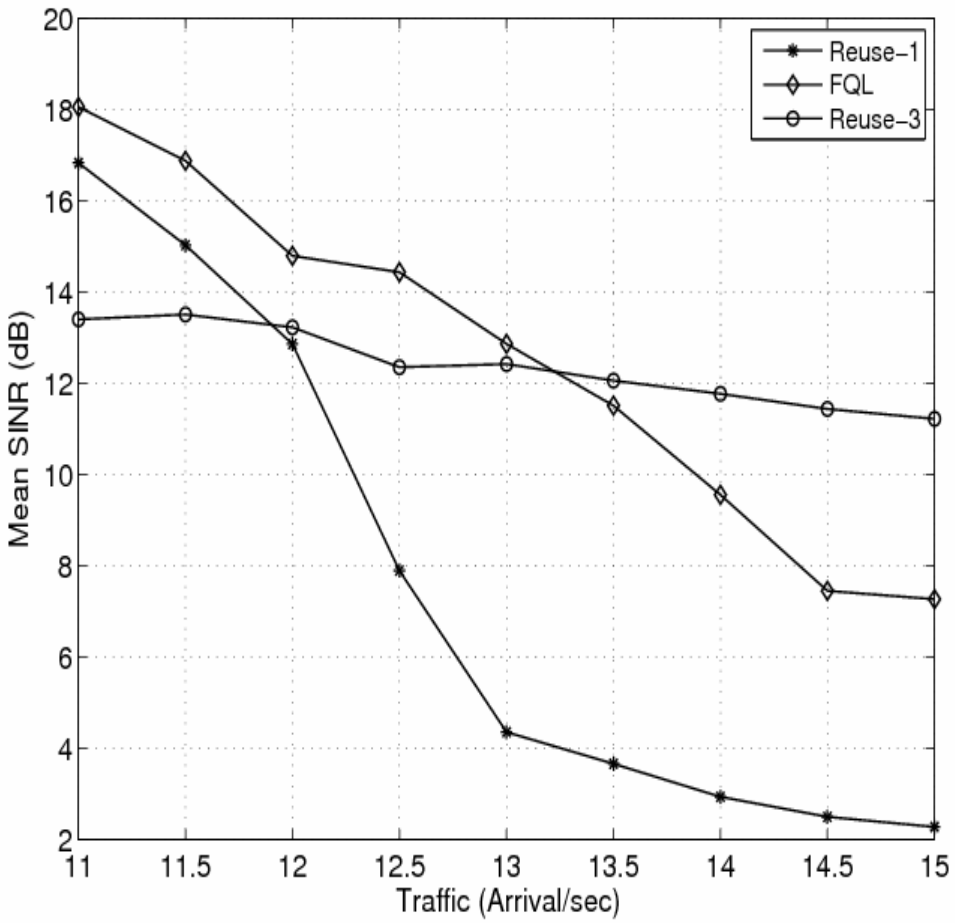
Why does it work ?

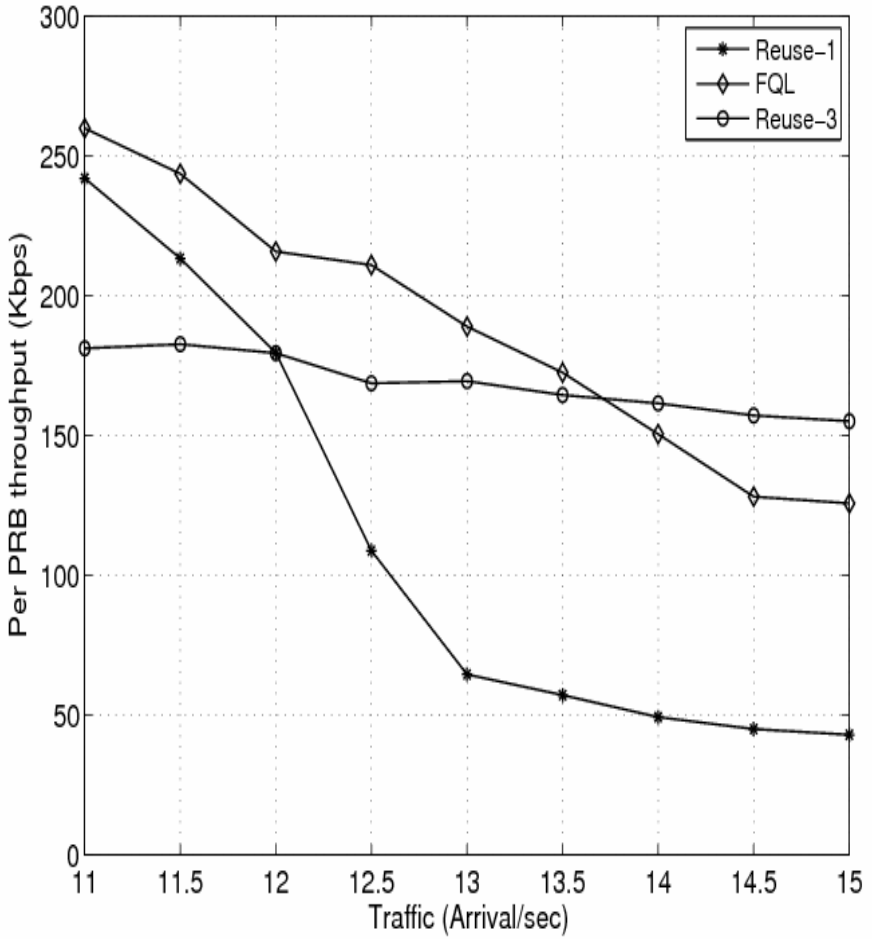
1. Served traffic (higher bound) is limited by the CAC
2. Learning phase covers low and high traffic states
3. Cooperative learning phase mixes experience of different cells
4. However, if the operating conditions are very different, learning in these conditions may be necessary

Results

Reuse-3



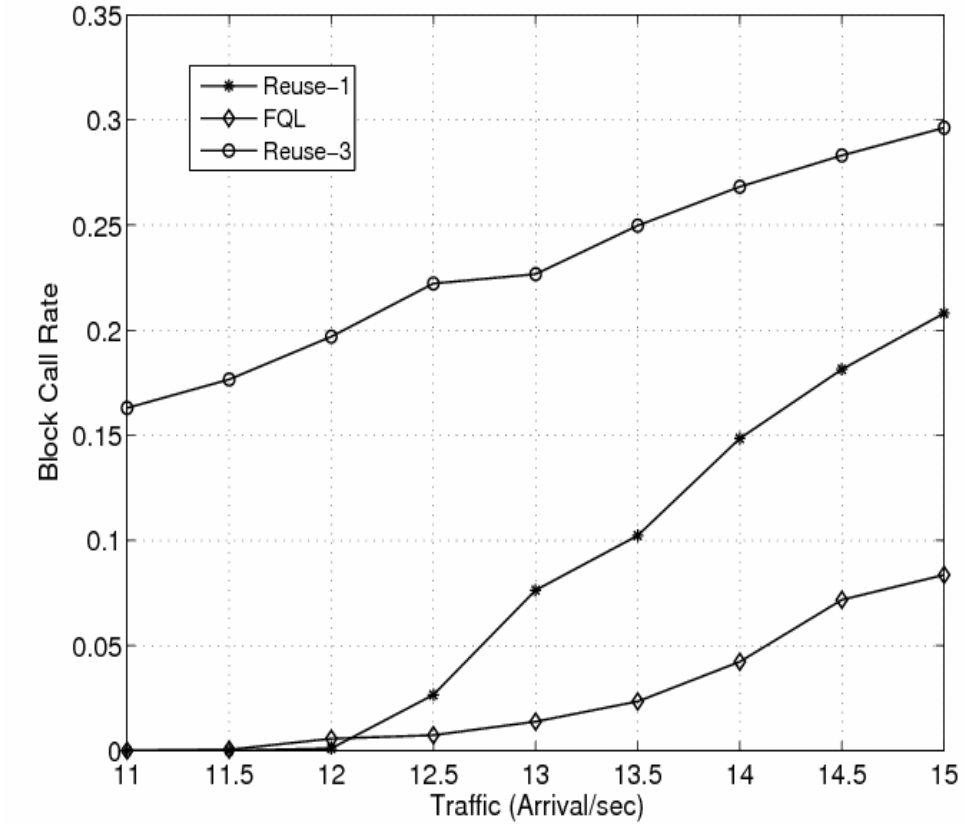


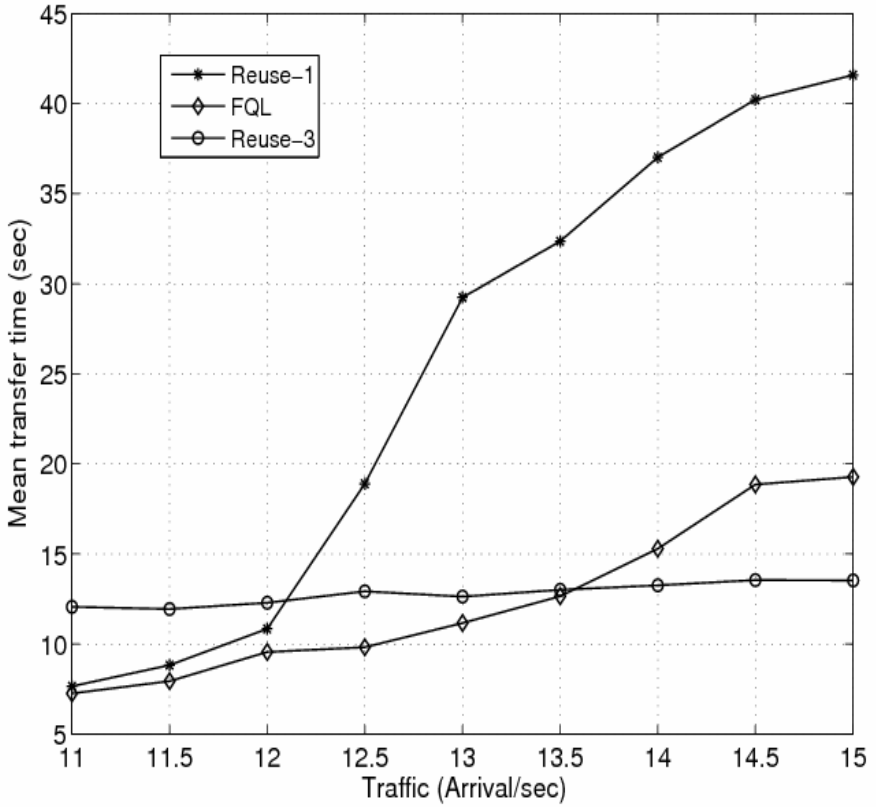


M. Dirani, Z. Altman, "A cooperative Reinforcement Learning approach for inter-cell interference coordination in OFDMA cellular networks", submitted

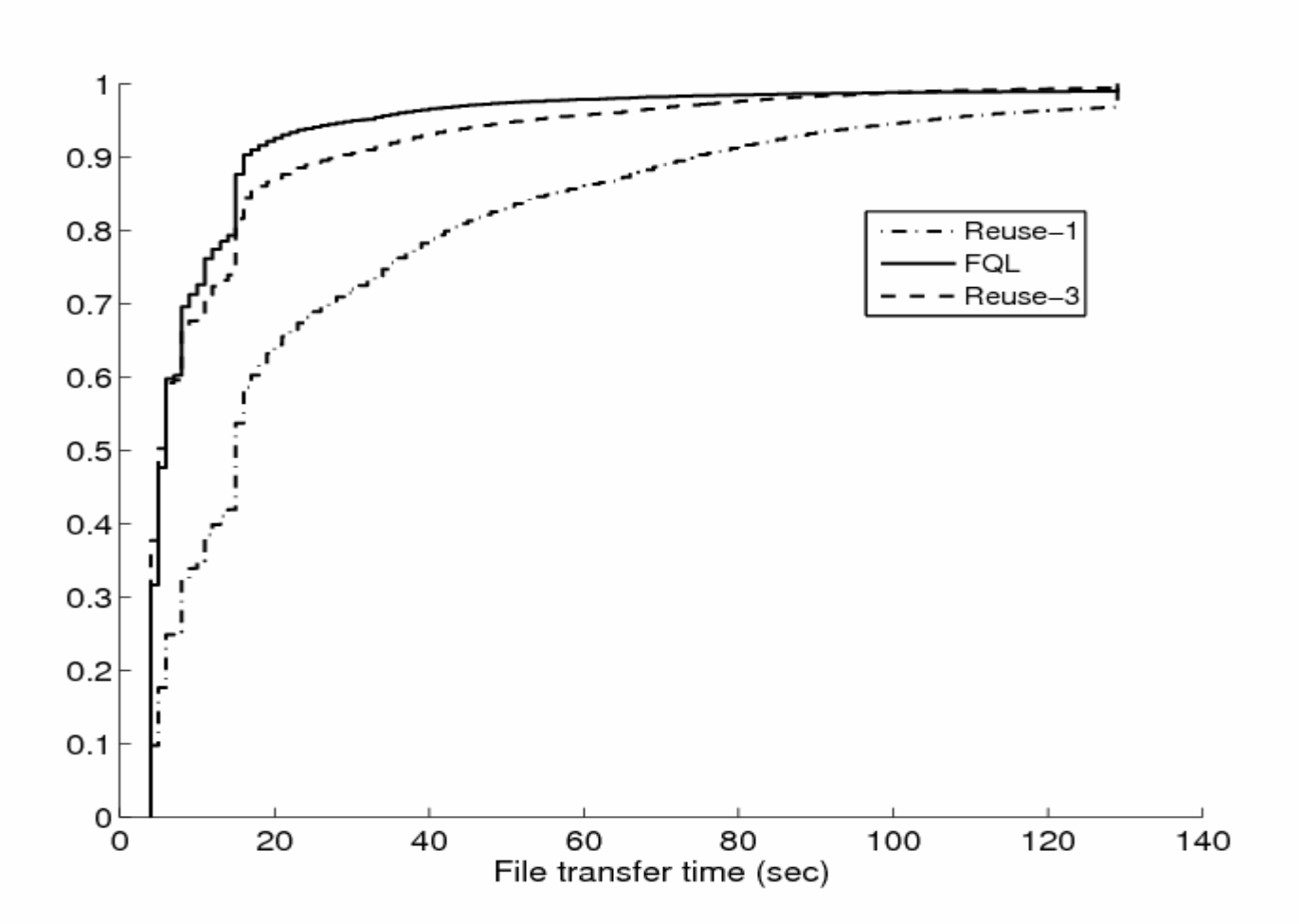
orange™

Results


Reuse–1
FQL
Reuse–3
Block Call Rate
Traffic (Arrival/sec)
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
11
11.5
12
12.5
13
13.5
14
14.5
15

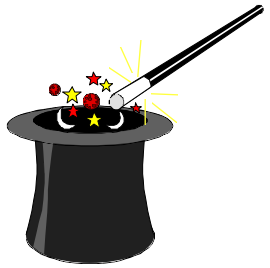

Reuse–1
FQL
Reuse–3
Mean transfer time (sec)
Traffic (Arrival/sec)
45
40
35
30
25
20
15
10
5
11
11.5
12
12.5
13
13.5
14
14.5
15

Results


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
20
40
60
80
100
120
140
File transfer time (sec)
Reuse–1
FQL
Reuse–3

Conclusions

- Reinforcement Learning provides a robust and simple framework for designing self-optimizing functions
  - Simple to implement
  - Robust with respect to noise introduced by traffic and propagation fluctuations



- Future directions / challenges
  - Simplified learning algorithms to implement in real networks
  - Further distribution of the control to allow coordinated operation of SON entities
    - Game/team theory approaches