Association problem in wireless networks : Policy Gradient Reinforcement Learning approach

Richard Combes, Ilham El Bouloumi, Stephane Senecal, Zwi Altman

Orange Labs

Ecole d'été de Peyresq 2013, 28th June 2013

Agenda

Introduction : Association Problem

- Mobile networks more and more heterogeneous $\rightarrow$ network operator needs to manage different RAN technologies : GSM, HSDPA, LTE, WLAN, ...
- Connection to network with lower load via advanced RRM algorithms (implementing mobility and selection/re-selection mechanisms) can impact significantly network performance and perceived QoS ;
- RRM burden shifts from network to UE, which can learn how to take smart association decision ;
- To which Base Station (BS) a UE should connect to in order to optimize network performance ? $\rightarrow$ *Association Problem*

Association problem : SON (1/2)

- Introduction of Self-Organizing Networks (SON) in 4G mobile networks ;
- Intra-system mobility load balancing optimization = self-optimization feature introduced in LTE standard ;
- Self-optimization aims at adapting network to traffic variations and to new conditions of operations ;
- Self-optimization process can be performed by autonomously adjusting network parameters (e.g. parameters of RRM algorithms) ;
- Real operating networks introduce strict requirements : scalability and stability.

Association problem (2/2)

- Scalability : SON features should operate correctly when deployed in many network nodes (BS and neighboring ones) ;
- Stability : network empowered by SON functionnality diminishes congestion in the network $\rightarrow$ number of active users remains bounded and tends to a stationary regime ;
- Deriving optimal parameters values or controllers via learning (*Reinforcement Learning*) $\Rightarrow$ learning or exploration phase $\rightarrow$ monotonic performance improvement during learning phase (*robust learning*) ;
- Our contribution : development of a self-optimized association algorithm based on *Policy Gradient Reinforcement Learning*, which is both scalable, stable and robust.

Reinforcement Learning (1/2)

- Association problem modeled as a Markov Decision Process (MDP) ;
- Requirement of robustness $\Rightarrow$ exclusion of direct application of RL solutions like Q-Learning ;
- Value Iteration cannot be applied as no a priori knowledge of the system dynamics (transition probabilities of the MDP) is available ;
- Optimal policy for the MDP is derived for a small size problem $\Rightarrow$ learn a functional form and parameterize the policy space ;
- Obtained solution used as *expert knowledge* for the Policy Gradient Reinforcement Learning algorithm.

Reinforcement Learning (2/2)

- Policy Gradient Reinforcement Learning algorithm converges to a local optimum and the average cost decreases monotically during the learning phase $\Rightarrow$ good candidate for practical implementation ;
- Robustness $\Rightarrow$ use the Policy Gradient Reinforcement Learning method in an "always-on" learning mode.

Agenda

Association Problem (1/3)

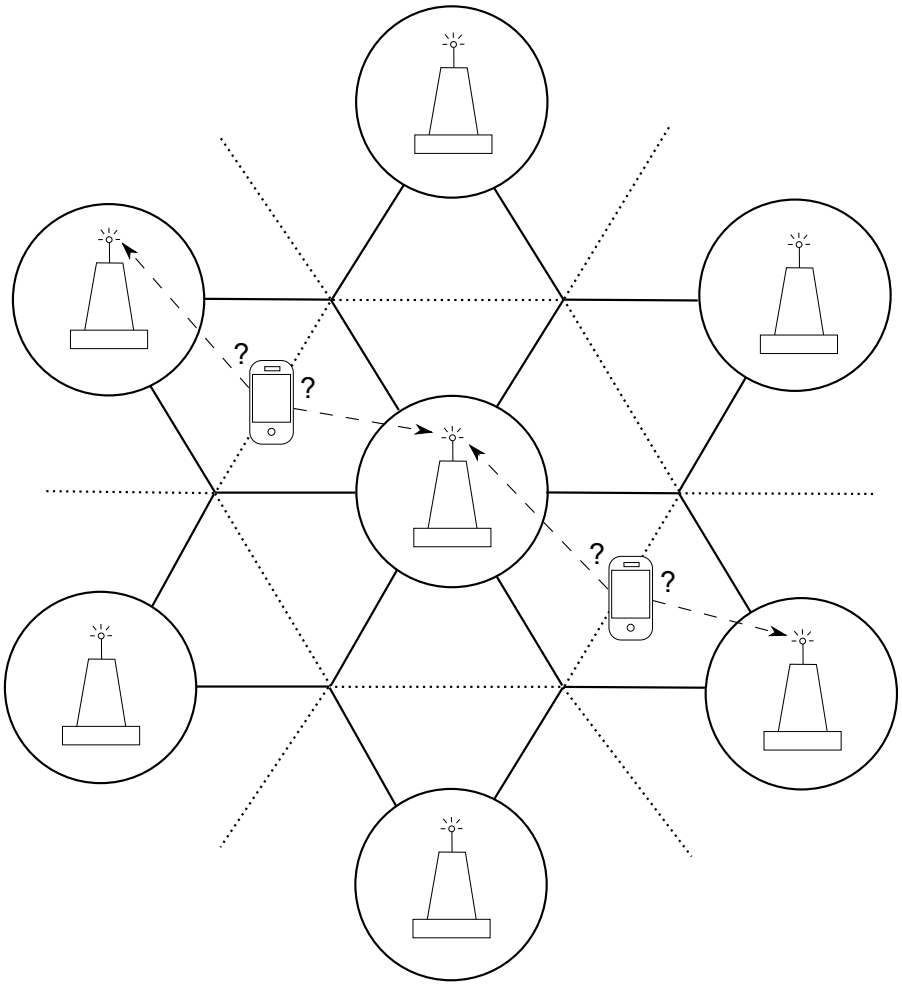


Figure: Association problem in the simplifed setting.

Association Problem (2/3)

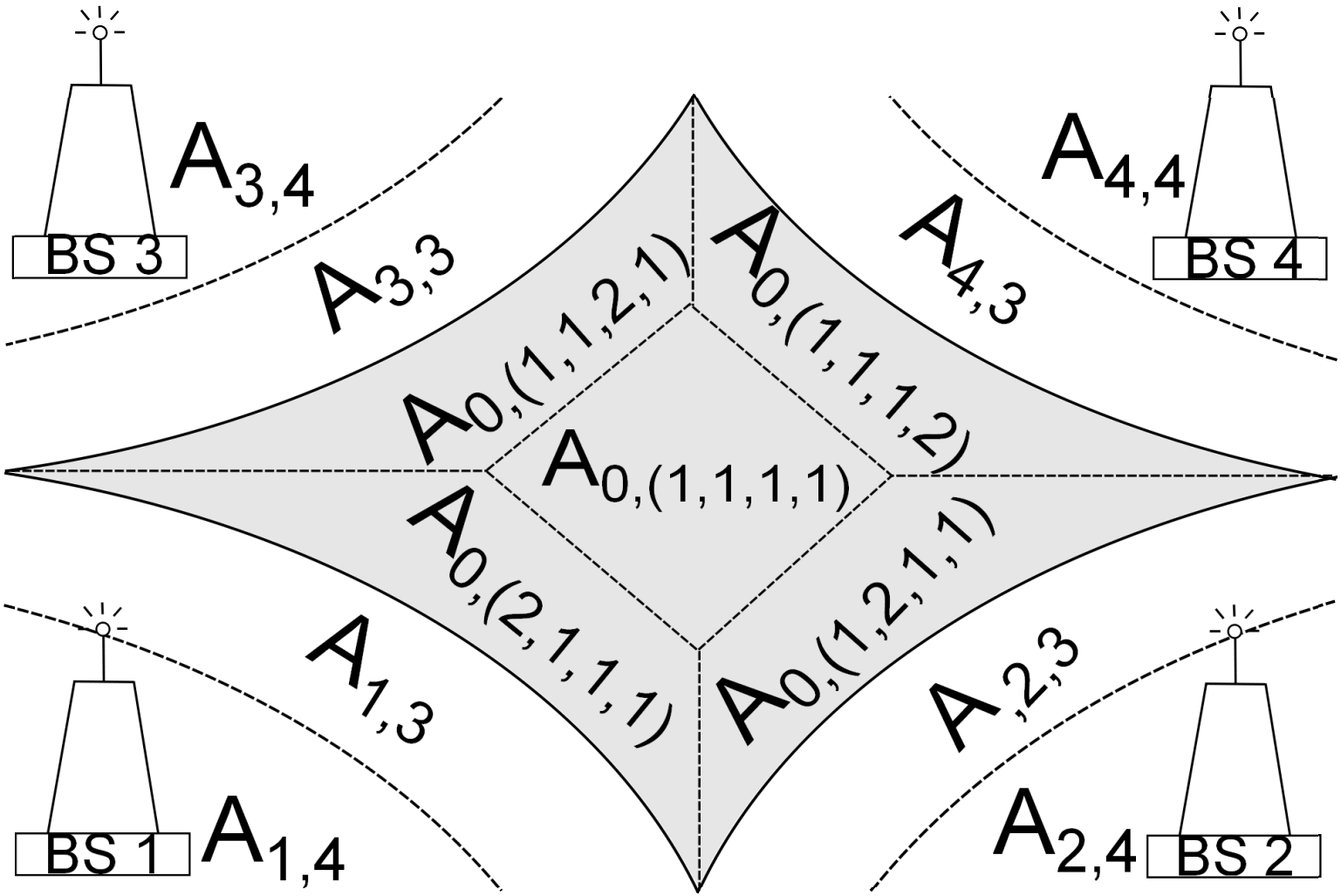


Figure: Association problem $\rightarrow$ determine the proportion of traffic from $A_{0,I}$ to be served by BS s, for all s and I.

Association Problem (3/3)

- Let us consider a new zone $A_0 \subset A$ in the network ;
- Association problem $\Rightarrow$ allocate the traffic arriving in A_0 to BSs in order to optimize a given performance indicator ;
- Dynamic problem : system has access to the current user configuration to make a decision ;
- User configuration composed of the number of active users, their locations, the BS they are currently attached to and their remaining amount of data to be downloaded ;
- Policy = mapping between user configuration and association of each user to a BS ;
- Problem consists in finding the policy that maximizes a given performance indicator $\rightarrow$ MDP modeling.

Markov Decision Process (1/2)

- Discrete state $s \in S$ and action $a \in A$ of the system ;
- Transition $P(s, a, s')$ and expected reward $R(s, a)$ models ;
- $\Rightarrow$ Find policy $\pi : s \in S \mapsto \pi(a|s)$
  in order to optimize an objective/target function $f_\pi(R(s, a))$

Markov Decision Process (2/2)

- Average cost of policy π starting at $s \in S$ can be defined by

$$J_\pi(s) = \limsup_{T \to +\infty} \frac{1}{T} E_\pi \left\{ \int_0^T r_t dt \right\} \tag{1}$$

$$a_t \sim \pi(.|s_t) \tag{2}$$

$$r_t \sim R(s_t, a_t) \tag{3}$$

- π s.t. $\{s_t\}$ ergodic $\Rightarrow$ $s \mapsto J_\pi(s)$ constant : $J_\pi(s) = J_\pi$;
- Solving the MDP $\Rightarrow$ find the optimal policy which minimizes the cost

$$\pi^\star = \arg\min_\pi J_\pi \tag{4}$$

MDP Modeling (1/2)

- Association problem can be modeled as a continuous time MDP ;
- Let us write $n_{s,l}$ the number of users of class $(0, l)$ that are attached to BS s ;
- The user configuration (state s with previous notations) is

$$\mathbf{n} = (n_{s,l})_{1 \leq s \leq N_s, l \in \mathcal{I}} \tag{5}$$

which completely specifies the number of users of each class, and the BSs they are attached to (BS s belonging to a set of N_s BSs).

MDP Modeling (2/2)

- Cost of a state chosen as the total number of users in this state ;
- Ergodicity of policies and Little's Law $\Rightarrow$ J_π divided by the total arrival rate is in fact the mean file transfer time under the policy π ;
- Alternatively, we can define the cost to be 1 if at least one user has a throughput smaller than a target data rate, and 0 otherwise ;
- J_π is then the outage probability under the policy π.

Reinforcement Learning problem

- Derive the optimal policy without the knowledge of the probabilistic structure of the model, through trial-and-error ;
- Namely the transition matrix $P(s, a, s')$ and the distribution of the costs $R(s, a)$ are unknown ;
- We can only obtain and exploit MDP realizations $\{s_t, a_t, s_{t+1}, r_t\}$;
- $\rightarrow$ Simulation-based methods or model-free methods.

Agenda

Resolution Techniques and Scalability (1/2)

- With discretization and "uniformization" schemes, always possible to reduce a continuous time MDP to a discrete time MDP
  $\rightarrow$ in the following, for reinforcement learning, the discrete time version of the system is considered ;
- When the transition matrix $P(s, a, s')$ and the distribution of the costs $R(s, a)$ are known, and both S and A are finite, the optimal policy $\pi^\star$ can be derived via an iterative scheme, namely dynamic programming, thanks to a fixed-point relation holding at optimality ;
- In practice however, for large state spaces, this becomes numerically intractable ("curse of dimensionality").

Resolution Techniques and Scalability (2/2)

- Scalable approach $\rightarrow$ introduce a parameterized family of policies $\{\pi_\theta, \theta \in \Theta\}$ and define the cost $J(\theta) = J_{\pi_\theta}$ (using the previous hypothesis of ergodicity) ;
- This parameterization is a powerful idea for solving optimal control problems numerically with state spaces of large dimension ;
- Problem becomes the optimization of $\theta \mapsto J(\theta)$, which is assumed to be computationally tractable ;
- Note that the performance of such a scheme highly depends on the goodness of the chosen family of policies ;
- Choosing a good parameterization generally implies having some knowledge on the structure of the optimal controller $\rightarrow$ can be seen as a form of "expert knowledge".

Policy Gradient Reinforcement Learning (1/3)

- How to optimize $\theta \mapsto J(\theta)$, without the knowledge of the probabilistic structure of the model ?
- Assume that we are at least able to simulate the system for a fixed value of θ, and that $J(\theta)$ can then be computed by averaging the observed cost for a sufficiently long simulation ;
- Interested in both local and global optima.
- For global optima, since the problem is not convex in general, a search heuristic (e.g. genetic algorithm, particle swarm optimization) is needed, which requires to compute $J(\theta)$ for a large number of values of θ ;
- For local optima, we can use a descent method by computing $\nabla_\theta J(\theta)$.

Policy Gradient Reinforcement Learning (2/3)

- Crudest approach is to approximate the gradient using finite differences ;
- Computation of its k-th component by :

$$\frac{J(\theta + \varepsilon e_k) - J(\theta - \varepsilon e_k)}{2\varepsilon} \tag{6}$$

  for a small ε, where e_k stands for the k-th unit vector ;
- Possible whenever $J(\theta)$ can be computed, but requires a number of simulations equal to twice the number of components of θ ;
- Also, this approach is not suitable for an "on-line" implementation where instead of simulating the system, the algorithm must compute an estimate of the gradient based on observations from a real network.

Policy Gradient Reinforcement Learning (3/3)

- Approach proposed $\rightarrow$ estimate $\nabla_\theta J(\theta)$ from a single (discrete-time) sample path $\{s_t, a_t, s_{t+1}, r_t\}$;
- It is possible to compute iteratively the eligibility traces :

$$z(t+1) = \beta z(t) + \frac{\nabla_\theta \pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)} \tag{7}$$

and the related gradient estimates :

$$\Delta(t+1) = \Delta(t) + \frac{1}{t+1}(r_{t+1} z(t+1) - \Delta(t)) \tag{8}$$

- The gradient estimates converge almost surely to an ascent direction :

$$\liminf_{t \rightarrow +\infty} < \Delta(t), \nabla_\theta J(\theta) > > 0 \tag{9}$$

Rationale behind the parameterization choice (1/2)

- Let $\theta \in \mathbb{R}^{N_s \times (\mathcal{I}+1) \times |\mathbb{I}|}$ be a vector of weights ;
- When a user arrives in zone $A_{0,l}$, he/she has to evaluate, for each possible BS, the peak rate available with this BS, and the load of the BS, which depends on the number of active users already attached to the BS and their peak rate ;
- In order to take a decision, for each BS s, we compute the weighted sum

$$\theta_{s,0,l} + \sum_{1 \leq i \leq \mathcal{I}} \theta_{s,i,l} \, T_{s,i}(\mathbf{n}) \tag{10}$$

- The term $\theta_{s,0,l}$ is independent of the load, and is linked to the peak rate available at BS s, irrespective of its load ;
- The term $\sum_{1 \leq i \leq \mathcal{I}} \theta_{s,i,l} \, T_{s,i}(\mathbf{n})$ is load dependent and is a weighted sum of the number of active users with different peak rates.

Rationale behind the parameterization choice (2/2)

- The attachment rule should assign a positive probability to *all* possible decisions $\rightarrow$ the average cost should be differentiable with respect to θ ;
- When a user of class $(0, I)$ enters the network, he/she is attached to BS s with probability $p_{s,I}$:

$$p_{s,I}(\mathbf{n}, \theta) = \frac{\exp(\theta_{s,0,I} + \sum_{1 \leq i \leq \mathcal{I}} \theta_{s,i,I} T_{s,i}(\mathbf{n}))}{\sum_{1 \leq s \leq N_s} \exp(\theta_{s,0,I} + \sum_{1 \leq i \leq \mathcal{I}} \theta_{s,i,I} T_{s,i}(\mathbf{n}))} \tag{11}$$

- The policy space contains several "intuitively" good policies ;
- Note that the action rule proposed above is a smooth approximation to the maximum function.

Parameterized Policies

We give four policies which should perform well, at least from an intuitive point of view :

- Join the station offering the best peak rate ;
- Join the station offering the best data rate ;
- Join the station with the smallest workload ;
- Join the station with the shortest queue.

Those policies often appear as solutions of control problems of queueing systems.

Distributed Implementation (1/5)

- Each BS s has a central zone in which all users are attached to s, and for each neighboring BS s' there is a zone in which users can only be attached either to s or $s' \rightarrow I_{s,s'}$ zone ;
- Parameters which control the decisions for zone $I_{s,s'}$ are

$$(\theta_{s,i,I_{s,s'}}, \theta_{s',i,I_{s,s'}})_{0 \leq i \leq \mathcal{I}} \tag{12}$$

- To alleviate notation, we use the notation $\theta_{s,s'}$ to denote the parameters used in the decision of association to s or s'.

Distributed Implementation (2/5)

- Let $\Delta_{s,s'}$ and $z_{s,s'}$ be the components of Δ and z respectively relative to $\theta_{s,s'}$;
- Gradient estimation procedure equations (7) and (8) :

$$z_{s,s'}(t+1) = \beta z_{s,s'}(t) + \frac{\nabla_{\theta_{s,s'}} \pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)} \tag{13}$$

$$\Delta_{s,s'}(t+1) = \Delta_{s,s'}(t) + \frac{1}{t+1}(r_{t+1} z_{s,s'}(t+1) - \Delta_{s,s'}(t)) \tag{14}$$

- Algorithm distributed $\rightarrow$ actions are taken based on locally available information ;
- When a user arrives in the network and could be attached to BS s, then BS s needs only to know the number of active users in its neighboring BSs ;
- $z_{s,s'}(t)$ is computed solely based on the number of active users in BSs s and s'.

Distributed Implementation (3/5)

- Computation of $\Delta_{s,s'}(t)$ requires to know the costs r_t, which are not a local information ;
- For instance, if the cost is the number of active users in the whole network (case of minimization of the mean file transfer time), every BS needs to be aware of the number of active users in every BS in the network ;
- When the number of BSs grows, the gradient estimates become more noisy $\rightarrow$ random fluctuations of the costs in all BSs will affect the estimation of the gradient with respect to $\theta_{s,s'}$, although this parameter mainly impacts BSs s and s' ;
- Serious impairments for practical application
  $\Rightarrow$ suggestion of a heuristic.

Distributed Implementation (4/5)

- Let assume that the cost is a sum of "local costs" $r = \sum_{s=1}^{N_s} r^{(s)}$, one per BS ;
- For instance if the cost is the total number of active users in the network, the cost of BS s is simply the number of active users in BS s ;
- Heuristic proposed for the computation of the gradient with respect to $\theta_{s,s'} \rightarrow$ use only the local rewards for BSs s and s' :

$$z_{s,s'}(t+1) = \beta z_{s,s'}(t) + \frac{\nabla_{\theta_{s,s'}} \pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)} \tag{15}$$

$$\Delta_{s,s'}(t+1) = \Delta_{s,s'}(t) + \frac{1}{t+1}((r_{t+1}^{(s)} + r_{t+1}^{(s')}) z_{s,s'}(t+1) - \Delta_{s,s'}(t)) \tag{16}$$

Distributed Implementation (5/5)

- Heuristic fully distributed : $\Delta_{s,s'}$ can be computed solely based on the local costs $r^{(s)}$ and $r^{(s')}$;
- The intuitive explanation behind the noise reduction is that using the heuristic, any random fluctuation of the local cost in a BS which is far away from BS s will not affect the estimation of the gradient with respect to $\theta_{s,s'}$;
- This is merely a heuristic since we cannot guarantee that the gradient estimate will be a valid ascent direction at each step ;
- However, it performs very well numerically, and yields a considerable improvement of the gradient estimation accuracy (by a factor of 10, as shown hereinafter).

Agenda

Network

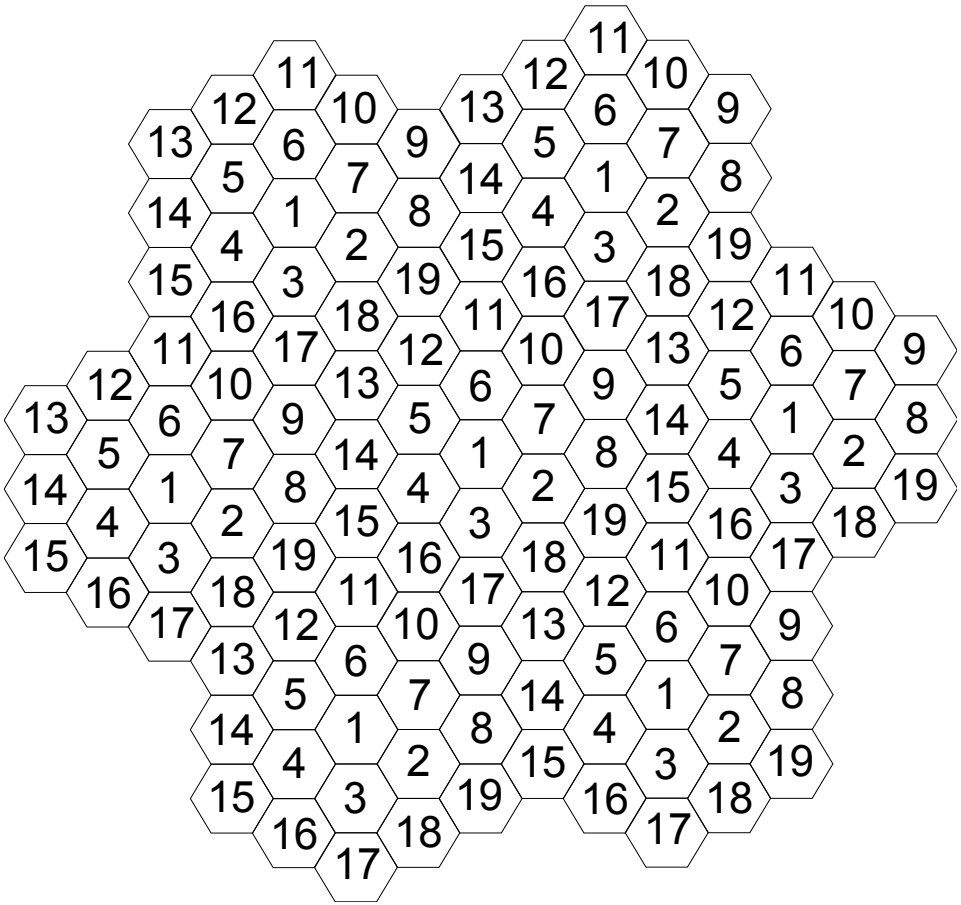


Figure: Hexagonal network of 19 BSs with wrap-around (to avoid border effects).

Traffic

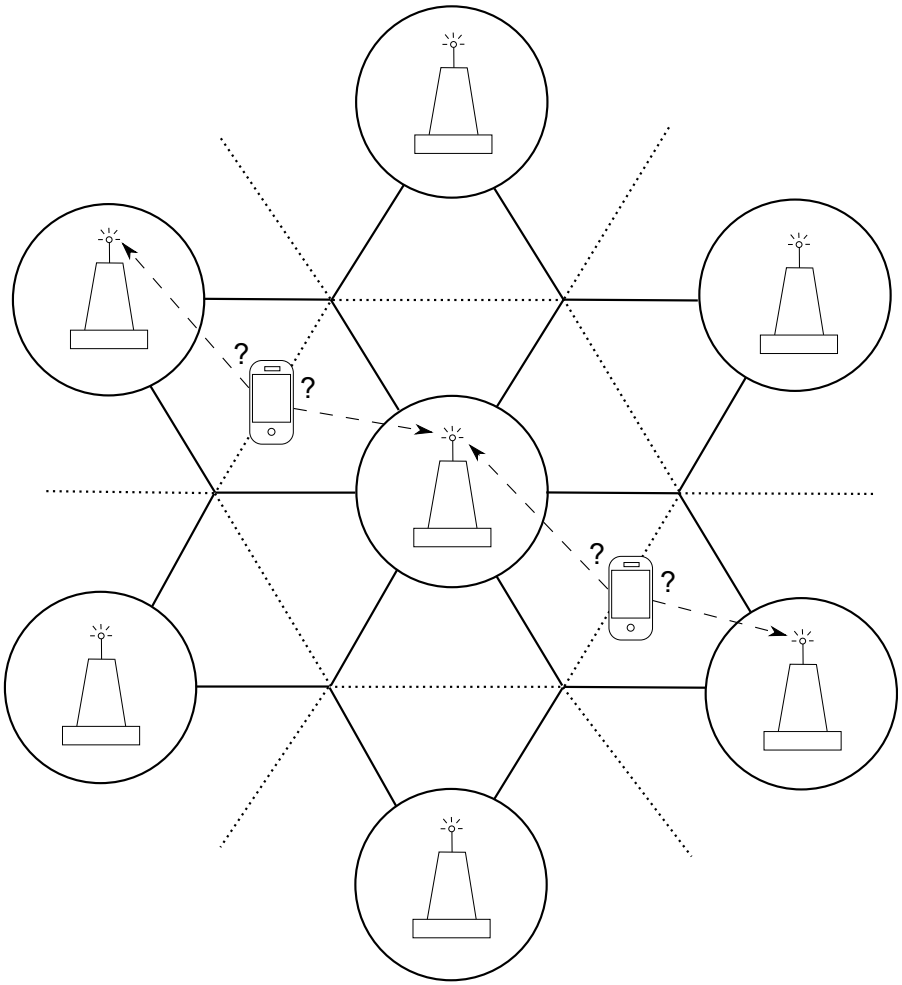


Figure: Association problem in the simplifed setting.

Setting

- Each BS s has a central zone where users are served with a data rate of 10Mbps ;
- The area of a central zone is $\frac{1}{2}$ of a cell area ;
- For each couple of BSs (s, s'), $s \neq s'$, there is a zone in which users can be served by either BSs s or s', both with a data rate of 5Mbps. The area of this zone is $\frac{1}{6}$ of a cell area, which is shared between BSs s and s' ;
- For the outage probability calculation, a target rate of 1Mbps is sought ;
- The mean file size is 10Mb.

Policies Comparison (1/3)

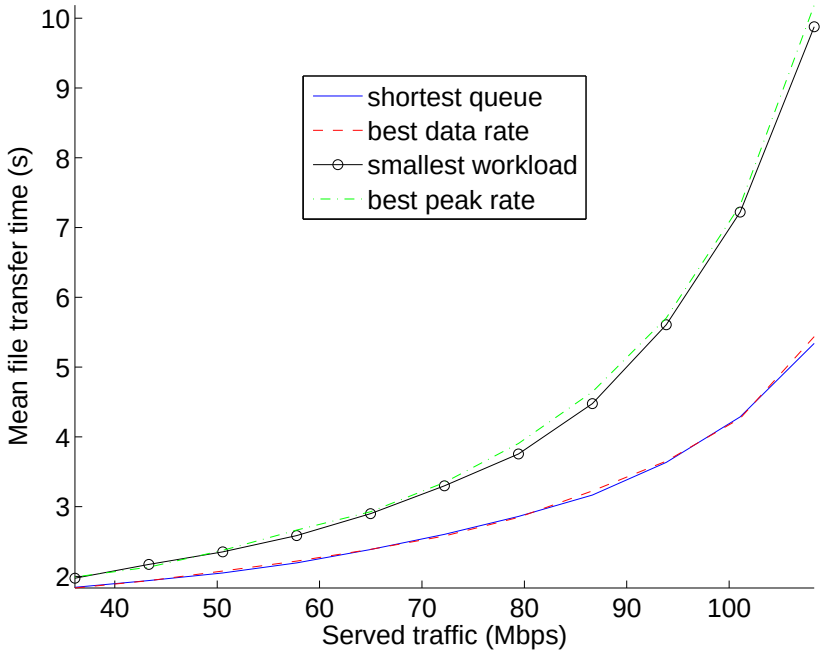


Figure: Mean file transfer time vs served traffic.

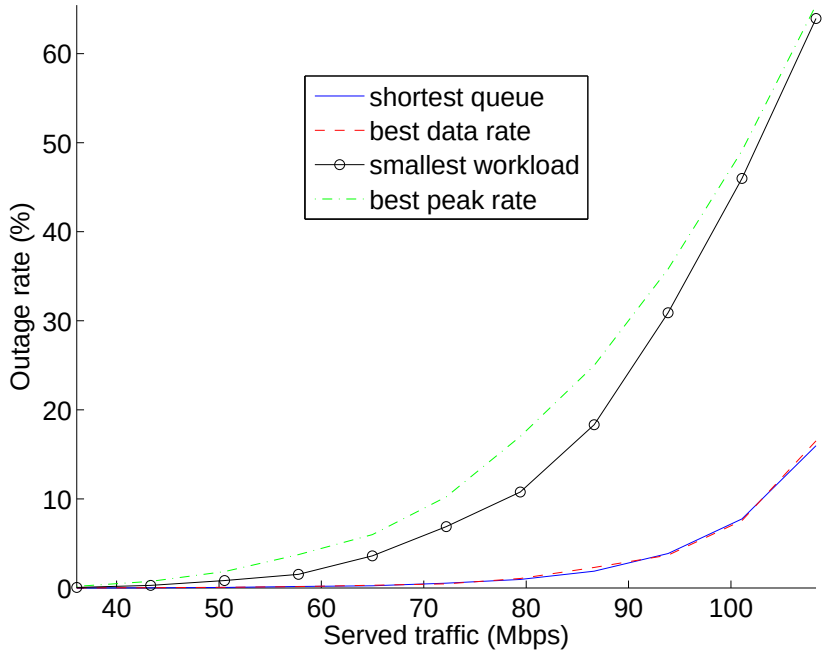


Figure: Outage probability vs served traffic.

Policies Comparison (2/3)

- Reference policy is "best peak rate" where users simply connect to the BS offering the best peak rate (i.e the best SINR), without considering the loads of available BSs ;
- Worst performance $\rightarrow$ not load-aware : it attaches users to the closest BS, even if it is overloaded, which does not reduce network congestion when traffic is high ;
- Policy "smallest workload" brings little improvement : even though it takes the loads into account, it can possibly admit a user when the number of active users is already large, resulting in outage ;
- $\rightarrow$ even for a large number of active users, the workload can be small if they have almost finished their transfer, or if their data rate is high.

Policies Comparison (3/3)

- Policies "best data rate" and "shortest queue" perform the best, and bring large improvement in both outage probability and mean file transfer time ;
- For high traffic, say 100Mbps, policies "best peak rate" and "smallest workload" yield a mean file transfer time of 7s and a outage probability of 60% ;
- Policies "best data rate" and "shortest queue" yield 4s for the mean file transfer time and 10% for the outage probability ;
- This shows that reducing congestion has a considerable impact on the network performance.

Accuracy of the Gradient Estimates (1/3)

- Accuracy of the gradient estimates obtained using the policy gradient standard method (denoted as "centralized") and the proposed heuristic which allows a distributed implementation (denoted as "distributed") ;
- Choice of $\theta = 0$ for comparison ;
- For a fixed number of time steps, generation of 500 gradient estimates using both methods ;
- Computation of the sign of the dot product between the gradient estimate and the true gradient obtained by finite difference for a long simulation with $100,000$ time steps ;
- If their dot product is strictly positive, then the gradient estimate is an admissible ascent direction ;
- Plot of the percentage of gradient estimates which are admissible ascent directions ;
- The higher the percentage, the better the gradient estimate is.

Accuracy of the Gradient Estimates (2/3)

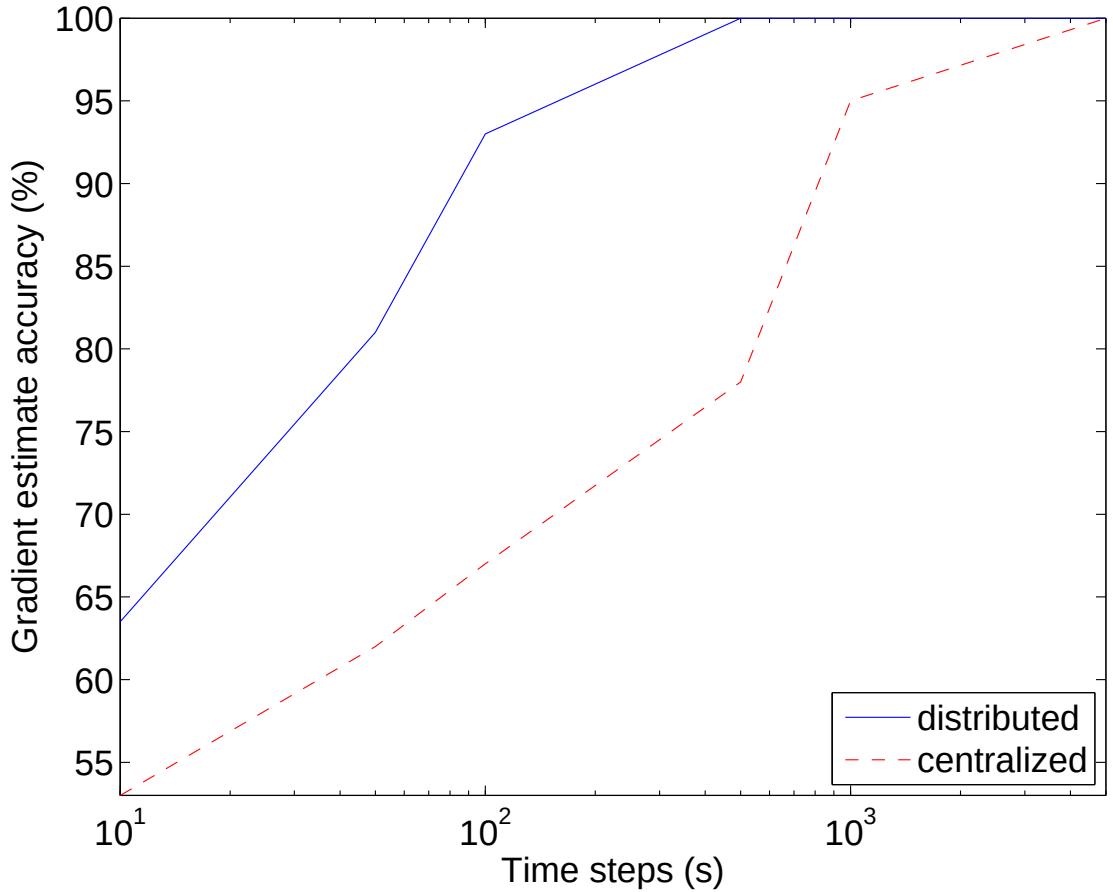


Figure: Accuracy of gradient estimates vs number of time steps.

Accuracy of the Gradient Estimates (3/3)

- Accuracy of gradient estimates goes to 100% when the number of time steps grows ;
- Proposed heuristic performs significantly better than the straighforward policy gradient ;
- For the same level of accuracy (say 95%) the number of time steps required by the heuristic is 10 times smaller than for the classical policy gradient.

Learning Process (1/3)

- Evolution of the average cost and of the corresponding controller parameter values during learning process ;
- Total traffic is 100Mbps ;
- Large number of parameters (57 in total) $\rightarrow$ only the two first components of the parameter vector are represented ;
- For each update of θ, the gradient is estimated during 100s.

Learning Process (2/3)

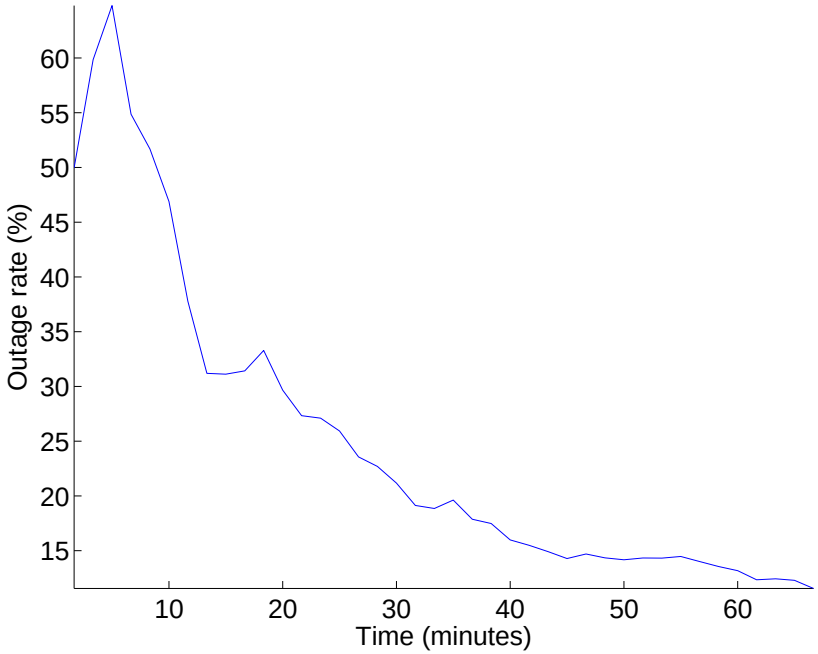


Figure: Average cost during the learning process.

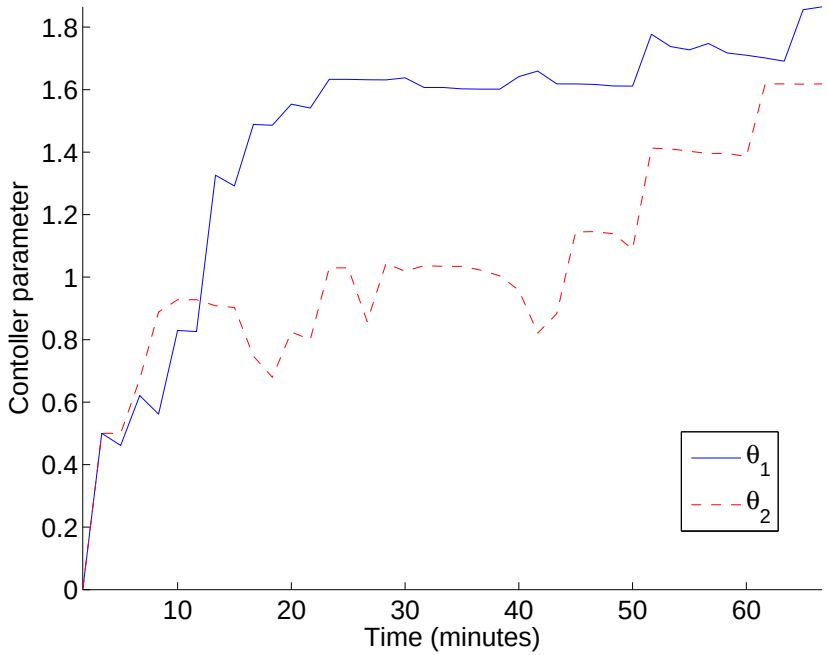


Figure: Controller parameters values during the learning process.

Learning Process (3/3)

- Starting from $\theta = 0$, and a heavily congested network, the outage probability diminishes almost monotonically ;
- $\Rightarrow$ the algorithm is able to find a configuration of parameters for which the congestion in the network is reduced ;
- Algorithm convergence speed with regards to the evolution of the daily traffic is satisfactory $\rightarrow$ in operational networks, the traffic pattern (arrival rates in each region) can reasonably be assumed fixed for at least one hour.

Agenda

Summary (1/2)

- Proposition of a model for association problem in wireless networks ;
- Taking into account traffic dynamics $\rightarrow$ optimize performance indicators directly perceived by users (mean file transfer time, outage probability) ;
- Modeling via Markov Decision Process (MDP) ;
- On-line *Policy Gradient Reinforcement Learning* method adapted to optimally control the system.

Summary (2/2)

- Introduction of a heuristic to enable a distributed implementation and improve the gradient estimation procedure ;
- The approach enjoys several advantages and thus is suitable for practical implementation :
  - Convergence to a local optimum ;
  - Average system performance improves monotically, and convergence speed is compatible with typical traffic evolution in operational networks ;
  - Solution scalable (complexity increases linearly with number of BSs) ;
  - Solution can be implemented in a distributed manner.
- Numerical experiments demonstrated that proposed solution performs well in practice, and effectively decreases congestion in the network.

References

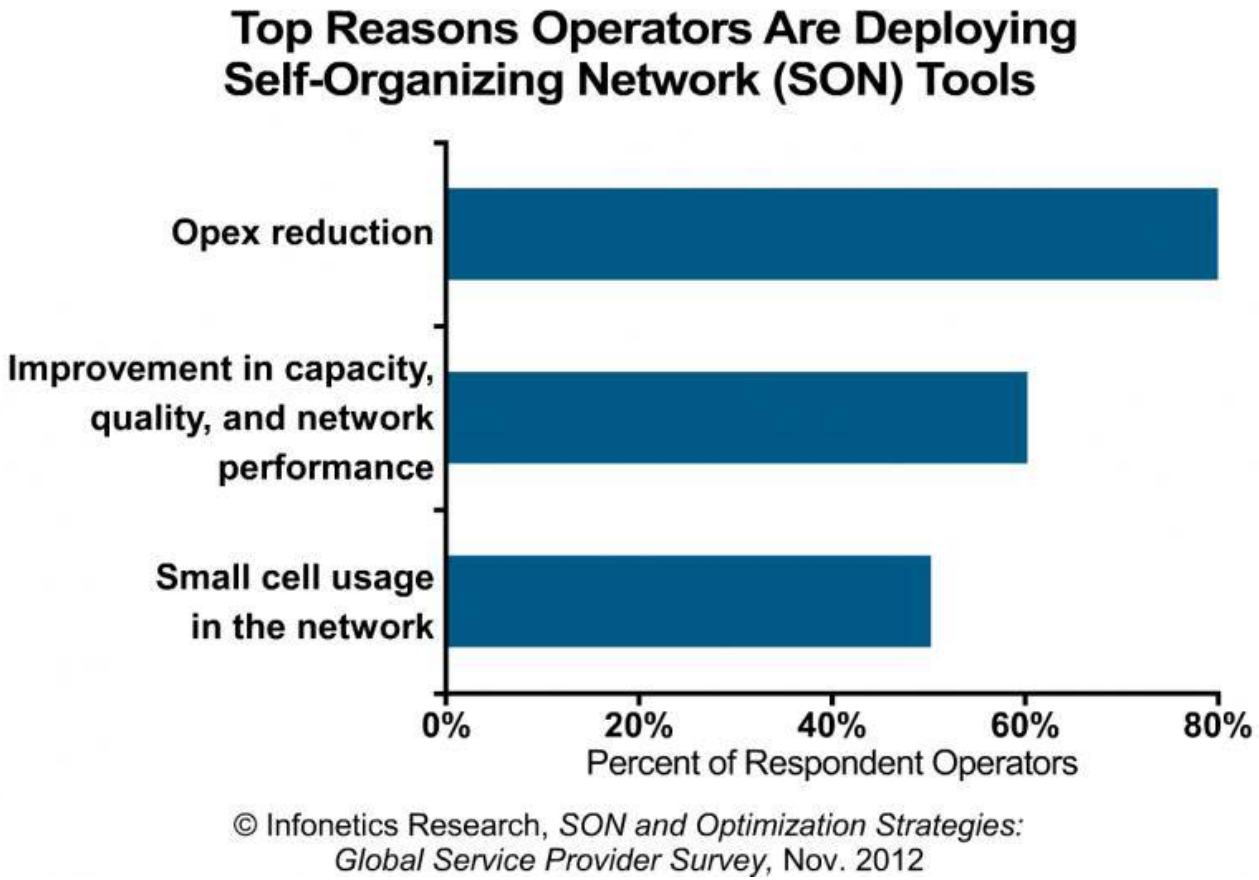
- R. Sutton and A. Barto, *Reinforcement Learning, An Introduction*, The MIT Press, 1998.
- R. J. Williams, *Simple statistical gradient-following algorithms for connectionist reinforcement learning*, Machine Learning, vol. 8, pp. 229-256, 1992.
- J. Baxter and P. L. Bartlett, *Infinite-Horizon Policy-Gradient Estimation*, Journal of Artificial Intelligence Research, vol. 15, pp. 319-350, 2001.
- J. Baxter, P. L. Bartlett and L. Weaver, *Experiments with Infinite-Horizon Policy-Gradient Estimation*, Journal of Artificial Intelligence Research, vol. 15, pp. 351-381, 2001.

Perspectives

- Problem : variance reduction for policy gradient estimates ;
- $\Rightarrow$ "Virtual Paths" techniques.

Thank You !

Appendix : SON



Appendix : Ergodicity

- Ergodicity : an irreducible, aperiodic, Harris-reccurent Markov chain which admits an invariant distribution is ergodic.
- Irreducibility : every non-zero probability set for distribution ϕ can be reached by the Markov chain in a finite number of steps/iterations for every starting point.
- Harris-recurrence : every non-zero probability set can be visited by the Markov chain infinitely often for every starting point.

Appendix : MDP Modeling

- We assume that association decisions are taken when users enter the system $\rightarrow$ avoid the possibility of constant hand-overs every time the user configuration changes, which would be impractical due to a high amount of additional overhead ;
- Note that the subset of states in which an action is available is relatively small, which is attractive in terms of practical controller implementation.

Appendix : Little's Law

- Little's Law : the long-term average number of customers in a stable system L is equal to the long-term average effective arrival rate λ multiplied by the (Palm-)average time a customer spends in the system W : $L = \lambda \times W$.
- $\Rightarrow$ This relationship holds for all arrival process distribution, service distribution, service order...

Appendix : Parameterized Policies

- The existence of those four policies has two practical implications :
- $\rightarrow$ finding the best parameterized policy yields performance at least as good as the previously described policies ;
- $\rightarrow$ if we seek to find the optimal value of θ through an iterative search (the optimal parameterized policy), for instance using gradient descent, then the initial value of θ can be chosen as one of those four policies ;
- This technique guarantees that even during the first iterations of the scheme, the system performance is already acceptable, as opposed to starting to a random value of θ which might yield very poor performance in the initial stages ;
- Note that when the number of users in BS s is significantly larger than in the other BSs, $p_{s,I}$ becomes very small and no users of A_0 are served by BS $s \Rightarrow$ parameterized family of policies introduces a form of load balancing.

Appendix : Setting

- Wrap-around essential : without, BSs on the outer ring would be significantly less loaded than BSs on the inner rings, and it would introduce considerable bias in the simulations ;