Optimization of a Fuzzy Logic Controller for Handover-based Load Balancing

P. Muñoz, R. Barco, I. de la Bandera, M. Toril and S. Luna-Ramírez
University of Málaga, Communications Engineering Dept., Málaga, Spain
Campus de Teatinos. 29071. Málaga
Email: {pabloml, rbm, ibanderac, mtoril, sluna}@ic.uma.es

Abstract—In Self-Organizing Networks (SON), load balancing has been recognized as an effective means to increase network performance. In cellular networks, cell load balancing can be achieved by tuning handover parameters, for which a Fuzzy Logic Controller (FLC) usually provides good performance and usability. Operator experience can be used to define the behavior of the FLCs. However, such a knowledge is not always available and hence optimization techniques must be applied in the controller design. In this work, a fuzzy Q-Learning algorithm is proposed to find the optimal set of fuzzy rules in an FLC for traffic balancing in GSM-EDGE Radio Access Network (GERAN). Load balancing is performed by modifying handover margins. Simulation results show that the optimized FLC provides a significant reduction in call blocking.

I. INTRODUCTION

Cellular networks have experienced a large increase in size and complexity in the last years. This has stimulated strong research activity in the field of Self-Organizing Networks (SONs) [1], [2].

A major issue tackled by SONs is the irregular distribution of cellular traffic both in time and space. To cope with such an increase in a cost-effective manner, operators of mature networks, such as GERAN, use traffic management techniques instead of adding new resources. One of such techniques is automatic load balancing, which aims to relieve traffic congestion in the network by sharing traffic between network elements. Thus, load balancing solves localized congestion problems caused by casual events, such as concerts, football matches or shopping centers.

In a cellular network, load balancing is performed by sharing traffic between adjacent cells. This can be achieved by changing the base station to which users are attached. When a user is in a call, it is possible to change the base station to which the user is connected by means of the handover (HO) process. By adjusting HO parameters settings, the size of a cell can be modified to send users to neighboring cells. Thus, the area of the congested cell can be reduced and that of adjacent cells take users from the congested cell edge. As a result of a more even traffic distribution, the call blocking probability in the congested cell decreases.

The study of load balancing in cellular networks can be found in the literature. In [3], an analytical teletraffic model for the optimal traffic sharing problem in GERAN is presented. In [4] the HO thresholds are changed depending on the load of the neighboring cells for real-time traffic. An algorithm to decide the balancing of the load between UMTS and GSM is presented in [5]. In this algorithm the decision for an intersystem HO depends on the load in the target and the source cell, the QoS, the time elapsed from the last HO and the HO overhead.

In the area of SON, [1] proposes a load balancing algorithm in a Long-Term Evolution (LTE) mobile communication system. This algorithm also requires the cell load as input and it controls the HO parameters. In this field, self-tuning has been applied to traffic management in many references. An UMTS/WLAN load balancing algorithm based on automatic adjustment of HO thresholds and cell load measurements for LTE systems is described.

All the previous references prove that FLCs can be successfully applied to automatic network parameter optimization. Their main strength is their ability to translate human knowledge into a set of basic rules. When an FLC is designed, a set of ‘IF-THEN’ rules must be defined, which represent the mapping of the input to the output in linguistic terms. Such rules might be extracted from operator experience. However, knowledge is not always available. In this case, reinforcement learning could be used to find the set of rules providing the best performance.

This paper investigates self-adaption of a fuzzy logic controller for load balancing in a cellular network. In particular, an FLC optimized by the fuzzy Q-Learning algorithm is proposed for traffic balancing of voice services in GERAN. It should be pointed out that GERAN is selected only as an example of radio access technology and results should be applicable to other radio access technologies. Unlike other FLCs optimized with reinforcement learning [6], the proposed controller modifies a specific parameter of the standard HO algorithm to reduce network blocking probability.

The rest of the paper is organized as follows. Section 2 describes the self-tuning scheme proposed. Section 3 presents the fuzzy Q-Learning algorithm. Section 4 discusses the simulation results and Section 5 presents the main conclusions of the study.
II. SELF-TUNING SCHEME

A. Handover parameters

In GERAN, Power BudGeT (PBGT) HO establishes that a HO is initiated whenever the average pilot signal level from a neighbor cell exceeds the one received from the serving cell by a certain margin, provided a minimum signal level is ensured, namely:

\[ R_{xLev} \geq R_xLevMin_{i\rightarrow j}, \quad (1) \]

\[ PBGT_{i\rightarrow j} = R_{xLev} - R_{xLev} \geq HOMargin_{i\rightarrow j}, \quad (2) \]

where \( R_{xLev} \) and \( R_{xLev} \) are the average received pilot signal levels from neighbor cell \( j \) and serving cell \( i \), \( PBGT_{i\rightarrow j} \) is the power budget of neighbor cell \( j \) with respect to serving cell \( i \), \( R_xLevMin_{i\rightarrow j} \) is the HO signal-level constraint and \( HOMargin_{i\rightarrow j} \) is the HO margin in the adjacency.

B. Fuzzy logic controller

The proposed FLC, shown in Fig. 1, is inspired in the diffusive load-balancing algorithm presented in [8]. The objective is to minimize call blocking ratio (CBR) in the network. For this purpose, FLCs (one per adjacency) compute the increment in HO margins for each pair of adjacent cells from the difference of CBR between these cells and their current HO margin values. By definition, the CBR is defined as:

\[ CBR = \frac{N_{blocked}}{N_{offered}} \quad (3) \]

where \( N_{blocked} \) is the number of blocked calls and \( N_{offered} \) is the total number of offered calls. Note that increments in the HO margins in both directions of the adjacency must have the same magnitude but opposite sign to provide a hysteresis region to ensure HO stability. Likewise, HO margins are restricted to a limited variation interval in order to avoid network instabilities due to excessive parameter changes. For simplicity, all FLCs are based on the Takagi-Sugeno approach [8]. The optimization block of the diagram is explained in the next section.

As in [9], one of the FLC inputs is the difference between the CBR of the two adjacent cells. Balancing the CBR between adjacent cells is achieved by iteratively changing HO margins on each adjacency. Thus, the overall CBR in the network is minimized, as shown in [3]. On the other hand, experience has shown that sensitivity to changes is greatly increased when margins becomes negative as a result of large parameter changes [7]. Thus, another input of the FLC is the current HO margin in order to reduce the magnitude of changes once margins become negative.

FLC inputs are classified according to linguistic terms by the fuzzyfier. The fuzzyfier translates input numbers into a value indicating the degree of membership to a linguistic term depending on a specific input membership function. For simplicity, the selected input membership functions are triangular or trapezoidal. In the inference engine, a set of IF-THEN rules are defined to establish a relationship between the input and the output in linguistic terms. Finally, the defuzzifier obtains a crisp output value by aggregating all rules. The output membership functions are constants and the centre-of-gravity method is applied to obtain the final value of the output.

III. FUZZY Q-LEARNING ALGORITHM

Reinforcement learning comprises a set of learning techniques based on leading an agent to take actions in an environment to maximize a cumulative reward. Q-Learning is a reinforcement learning algorithm where a Q-function is built incrementally by estimating the discounted future rewards for taking actions from given states. This function is denoted by \( Q(x, a) \), where \( x \) represents the states and \( a \) represents the actions. A fuzzy version of Q-Learning is used in this work due to several advantages. Firstly, it allows treating continuous state and action spaces, to store the state-action values and to introduce a priori knowledge.

In Fig. 1, the main elements of the reinforcement learning are represented. The set of environment states, \( S \), is given by the inputs of the FLC. The set of actions, \( A \), represents the FLC output, which is the increment of HO margins. The value \( r \) is the scalar immediate reward of a transition.

A discretization of the \( Q \)-function is applied in order to store a finite set of state-action values. If the learning system can choose one action among \( J \) for rule \( i \), \( a[i,j] \) is defined as the \( j^{th} \) possible action in rule \( i \) and \( q[i,j] \) is defined as its associated \( q \)-value. Hence, the representation of \( Q \)-function is built for each consequent of the rules, then to interpolate for an any input vector.

To initialize the \( q \)-values in the algorithm, the following assignment is used:

\[ q[i,j] = 0, \quad 1 \leq i \leq N \text{ and } 1 \leq j \leq J \quad (4) \]

where \( q[i,j] \) is the \( q \)-value, \( N \) is the number of rules and \( J \) is the number of actions per rule.

![Fig. 1. Block diagram of the fuzzy controller](image-url)
Once the initial state has been chosen, the FLC action must be selected. For all activated rules, with nonzero activation function, an action is selected using the following exploration/exploitation policy:

\[
a_i = \arg \max_k q[i, k] \quad \text{with probability } 1 - \epsilon
\]

\[
a_i = \text{random}\{a_k, k = 1, 2, \ldots, J\} \quad \text{with probability } \epsilon
\]

where \(a_i\) is the consequent of the rule and \(\epsilon\) is a learning rate which establishes the trade-off between exploration and exploitation in the algorithm (e.g., \(\epsilon = 0\) means that there is no exploration, that is, the best action is always selected). Then, the action to execute is determined by:

\[
a = \sum_{i=1}^{N} \alpha_i(x) \times a_i
\]

where \(a\) is the FLC action, \(\alpha_i(x)\) is the activation function for the rule \(i\) and \(a_i\) is the specific action for that rule. The activation function or degree of truth is the distance between the input state \(x(t)\) and the rule \(i:\)

\[
\alpha_i(x(t)) = \prod_{j=1}^{L} \mu_{ij}(x_j(t))
\]

where \(L\) is the number of FLC inputs, \(\mu_{ij}(x_j(t))\) is the membership function for the \(j^{th}\) FLC input and the rule \(i\).

The \(Q\)-function can be calculated from the current \(q\)-values and the activation functions of the rules:

\[
Q(x(t), a) = \sum_{i=1}^{N} \alpha_i(x) \times q[i, a_i]
\]

where \(Q(x(t), a)\) is the value of the \(Q\)-function for the state \(x(t)\) in iteration \(t\) and the action \(a\).

Subsequently, the system evolves and the new state \(x(t+1)\) and the reinforcement signal \(r(t+1)\) are observed. The FLC computes \(\alpha_i(x(t+1))\) for all rules, so that the value of the new state is calculated as:

\[
V_i(x(t+1)) = \sum_{i=1}^{N} \alpha_i(x(t+1)) \cdot \max_k q[i, a_k]
\]

The difference between \(Q(x(t), a)\) and \(Q(x(t+1), a)\) can be used to update the action \(q\)-values. This difference can be seen as an error signal and it is given by:

\[
\Delta Q = r(t+1) + \gamma \times V_i(x(t+1)) - Q(x(t), a)
\]

where \(\Delta Q\) is the error signal, \(r(t+1)\) is the reinforcement signal, \(\gamma\) is a discount factor, \(V_i(x(t+1))\) is the value of the new state and \(Q(x(t), a)\) is the value of the \(Q\)-function for the previous state. The action \(q\)-values can be immediately updated by an ordinary gradient descent:

\[
q[i, a_i] = q[i, a_i] + \eta \cdot \Delta Q \cdot \alpha_i(x(t))
\]

where \(\eta\) is a learning rate. Then, the above-described process is repeated starting with the action selection for the new current state.

To find the best consequents of the rules, the reinforcement signal must be selected appropriately. In this work, the reinforcement signal is defined as:

\[
r_p(t) = r_{p1}(t) + r_{p2}(t) + k_1
\]

where \(r_p(t)\) is the reinforcement signal for the \(p^{th}\) FLC in iteration \(t\), \(k_1\) is a constant parameter, and \(r_{p1}(t)\) and \(r_{p2}(t)\) are the reinforcement signal contributions of both cells in the adjacency, which are chosen as:

\[
r_{pi}(t) = k_2 \cdot \log\left(\frac{1}{(CBR_i + k_3) \cdot 1000 + 1}\right)
\]

where \(k_2\) and \(k_3\) are constant parameters and \(CBR_i\) is the call blocking ratio of the cell \(i\) in the adjacency.

Fuzzy \(Q\)-Learning algorithm aims to find the best consequent within a set of actions for each rule so as to maximize future reinforcements. The network operator may have different degrees of a priori knowledge for the rules, eventually being able to propose a set of consequents for a specific rule. The most unfavorable case is when there is no knowledge. In this case, consequents are equally distributed in the output interval. Another case is the imprecise knowledge, which occurs when the network operator is able to select a region more appropriate than others in the output interval. Lastly, if a rule can be exactly known, it is called precise knowledge.

As mentioned, the value of \(\epsilon\) determines the exploration/exploitation policy. During the exploration phase, the set of all possible actions should be evaluated to avoid local minima solutions. The convergence of the \(Q\)-Learning algorithm during the exploration phase is reached when the \(q\)-values becomes constant. In the exploitation phase, the best action with highest \(q\)-value is chosen most of the time. A small probability of experiencing other actions with less \(q\)-value is given by the policy if \(\epsilon\) is set to a small value. These actions may be worse at the moment, but eventually they may lead to zones with higher \(q\)-values.

## IV. Simulation

A dynamic GERAN system-level simulator has been developed in MATLAB. This simulator first runs a module of parameter configuration and initialization, where a warm-up distribution of users is generated. Then, a loop is started until the end of the simulation, where the update of user positions, propagation computation, generation of new calls, and radio resource management algorithms are executed. Finally, the main statistics and results of the simulation are shown.

The simulated scenario includes a macro-cellular environment whose layout consists of 19 tri-sectorized sites evenly distributed in the scenario. The main simulation parameters are summarized in Table I. Only the downlink is considered in the simulation, as it is the most restrictive link in GERAN.
For simplicity, the service provided to users is the circuit-switched voice call as it is the main service affected by the tuning process.

A non-uniform spatial traffic distribution is assumed in order to generate the need for load balancing. Thus, there are several cells with a high traffic density whereas surrounding cells have a low traffic density. It is expected that parameter changes performed by the FLCs manage to relieve congestion in the affected area. It is noted that about 15-20 iterations are sufficient to obtain useful \( q \)-values.

In Fig. 2, the reinforcement signal is represented as a function of the CBR. A low CBR is rewarded with positive reinforcement values whereas a high CBR is penalized with negative reinforcement values. A single look-up table of action \( q \)-values is shared by all FLCs, in such a way that the look-up table is updated by all FLCs at each iteration and, thus, the convergence process is accelerated.

Once the \( Q \)-Learning algorithm has been applied to the controllers, the best consequent for each rule is determined by the highest \( q \)-value in the look-up table. Table II shows the set of candidate actions selected for each rule from imprecise knowledge (4th column) together with the consequent obtained by the optimization process (5th column). The linguistic terms defined for the fuzzy input 'HO margin' are \( L \) (low), \( M \) (medium) and \( H \) (high), while those defined for the fuzzy output '\( \Delta \)HO margin' are \( EL \) (extremely low), \( VL \) (very low), \( L \) (low), \( N \) (null), \( H \) (high), \( VH \) (very high) and \( EH \) (extremely high), which correspond to the crisp output values -8, -4, -1, 0, 1, 2, 4 and 8 dB, respectively.

As shown in table II, rules 1 and 7 are triggered when the difference between CBRs is high and the current HO margin has a value which is opposite to the required value. The last column shows that the optimization process concludes that the best consequents for these rules are the largest modification in HO margins. Rules 2 and 8 are triggered when the difference between CBRs is high and the current HO margin has an intermediate value. In this case, the optimal actions are moderate changes in HO margins. Rules 3 and 9 are activated when there is a high difference between CBRs but the current HO margin belongs to the desired interval. In this situation, HO margins might experience saturation and the \( Q \)-Learning algorithm has some difficulty in finding the best action. It should be pointed out that the optimization process avoids large changes in HO margins because network sensitivity increases with negative HOs. Rules 4 and 6 are triggered when traffic is balanced and the current HO margin belongs to the desired extreme interval. Here, the optimal actions depend on the current load distribution in such a way that it can be more suitable to bring the handover margin to an intermediate value in order to return users to the original cells or it can be better to leave it at the same value. A combination between exploration and exploitation would be necessary here to determine the best action at any time. Finally, rule 5 is the only one that has been selected by precise knowledge because when the traffic is balanced and the HO margin has a neutral value, the best action is obviously the same value for the HO margin.

Fig. 3 shows an example of how the best consequent is selected for a rule by showing the \( q \)-value evolution of consequents in rules 1 and 2 during a simulation. It is observed that the consequent \( EL \) must be considered as the best action for rule 1, since it has the largest \( q \)-value across iterations. Regarding rule 2, the best consequent is \( VL \).

To check the influence of the parameter \( \epsilon \) into network performance, several simulations in the exploitation phase have been carried out. For this purpose, one FLC configuration

---

**TABLE I**

<table>
<thead>
<tr>
<th>Cellular layout</th>
<th>Hexagonal grid, 57 GSM cells, cell radius 0.5 km</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propagation model</td>
<td>Okumura-Hata with wrap-around correlated log-normal slow fading, ( \sigma_f=8 )dB</td>
</tr>
<tr>
<td>Mobility model</td>
<td>Random direction, constant speed 3 km/h</td>
</tr>
<tr>
<td>Service model</td>
<td>Speech, mean call duration 100s, activity factor 0.5</td>
</tr>
<tr>
<td>BS model</td>
<td>Tri-sectorized antenna, EIRP max=43dBm</td>
</tr>
<tr>
<td>Adjacency plan</td>
<td>Symmetrical adjacencies, 18 per cell</td>
</tr>
<tr>
<td>RRM features</td>
<td>Random FH, POC, DTX</td>
</tr>
<tr>
<td>Traffic distribution</td>
<td>Unevenly distributed in space</td>
</tr>
<tr>
<td>Time resolution</td>
<td>SACCH frame (480 ms)</td>
</tr>
<tr>
<td>Simulated time</td>
<td>4 h (per optimization epoch)</td>
</tr>
</tbody>
</table>

---

**TABLE II**

<table>
<thead>
<tr>
<th>Rule</th>
<th>CBR1-CBR2</th>
<th>HO Margin</th>
<th>Candidate actions</th>
<th>Best action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Unbalanced (\rightarrow) 2</td>
<td>H</td>
<td>EL, VL, L</td>
<td>EL</td>
</tr>
<tr>
<td>2</td>
<td>Unbalanced (\rightarrow) 2</td>
<td>M</td>
<td>EL, VL, L</td>
<td>VL</td>
</tr>
<tr>
<td>3</td>
<td>Unbalanced (\rightarrow) 2</td>
<td>L</td>
<td>VL, L, N</td>
<td>L</td>
</tr>
<tr>
<td>4</td>
<td>Balanced</td>
<td>H</td>
<td>VL, L, N</td>
<td>N</td>
</tr>
<tr>
<td>5</td>
<td>Balanced</td>
<td>M</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>6</td>
<td>Balanced</td>
<td>L</td>
<td>N, H, VH</td>
<td>VH</td>
</tr>
<tr>
<td>7</td>
<td>Unbalanced (\rightarrow) 1</td>
<td>L</td>
<td>H, VH, EH</td>
<td>EH</td>
</tr>
<tr>
<td>8</td>
<td>Unbalanced (\rightarrow) 1</td>
<td>M</td>
<td>H, VH, EH</td>
<td>VH</td>
</tr>
<tr>
<td>9</td>
<td>Unbalanced (\rightarrow) 1</td>
<td>H</td>
<td>N, H, VH</td>
<td>H</td>
</tr>
</tbody>
</table>

---

![Fig. 2. Reinforcement signal](image-url)
has been defined without exploration ($\epsilon = 0$), where the set of consequents obtained by the optimization process is fixed during the simulation. Another configuration considers exploration ($\epsilon = 0.8$) and the initial $q$-values have been set to 10 for the best consequents while the others have been set to 0. In this case, consequents are adapted dynamically by the $Q$-Learning algorithm to manage the load imbalance.

Fig. 4 shows the CBR for 19 selected cells distributed uniformly in the network. The row of bars at the back corresponds to the initial situation of load imbalance. It is observed that there are cells with high CBR and others with negligible CBR. The central row of bars corresponds to the end of a simulation when FLCs have been performed without exploration. It is clear that traffic load is now shared between cells and CBR is more equalized than the initial situation. Finally, the row of bars in front corresponds to the end of a simulation when FLCs have been performed with $\epsilon = 0.8$. As reflected in Fig. 4, load imbalance can be reduced even more if exploration is also applied.

The main drawback preventing operators from fully exploiting the potential of the HO-based load balancing is the increase in the Call Dropping Ratio (CDR). As expected, global CDR is slightly increased compared to the initial situation. Particularly, when exploration is considered ($\epsilon = 0.8$), there is an increase of 0.2% in the CDR. This value is similar to the case without exploration ($\epsilon = 0$). In this manner, $Q$-learning algorithm maintains the same performance in both cases.

V. CONCLUSIONS

This paper has described an optimized FLC tuning dynamically HO margins for load balancing in GERAN. The optimization process is performed by the fuzzy $Q$-Learning algorithm, which is able to find the best consequents for the rules of the FLC inference engine. The learning algorithm had some difficulties to distinguish between actions, in one case due to the saturation of the HO margin, and in another case due to the time-variant nature of the rule. In the latter case, it can be solved by providing some degree of exploration to the $Q$-Learning algorithm during the exploitation phase.

Simulation results show that the optimization process achieves a significant reduction in call blocking for congested cells, which lead to an acceptable decrease in the overall call blocking. The FLC can also be considered as a cost-effective solution to increase network capacity because hardware upgrade is not necessary. The main disadvantage is a slight increment in call dropping and an increase in network signaling load due to a higher number of HOAs.

ACKNOWLEDGMENT

This work has partially been supported by the Junta de Andalucía (Excellence Research Program, project TIC-4052) and by the Spanish Ministry of Science and Innovation (grant TEC2009-13413).

REFERENCES