A Call Admission Control Scheme using NeuroEvolution Algorithm in Cellular Networks

Xu Yang
MPI-QMUL
Information Systems Research Centre
A313, Macao Polytechnic Institute
Macao SAR, China
xuy@mpi-qmul.org

John Bigham
Department of Electronic Engineering
Queen Mary University of London,
London E1 4NS, U.K.
john.bigham@elec.qmul.ac.uk

Abstract

This paper proposes an approach for learning call admission control (CAC) policies in a cellular network that handles several classes of traffic with different resource requirements. The performance measures in cellular networks are long term revenue, utility, call blocking rate (CBR) and handoff failure rate (CDR). Reinforcement Learning (RL) can be used to provide the optimal solution, however such method fails when the state space and action space are huge. We apply a form of NeuroEvolution (NE) algorithm to inductively learn the CAC policies, which is called CN (Call Admission Control scheme using NE). Comparing with the Q-Learning based CAC scheme in the constant traffic load shows that CN can not only approximate the optimal solution very well but also optimize the CBR and CDR in a more flexibility way. Additionally the simulation results demonstrate that the proposed scheme is capable of keeping the handoff dropping rate below a pre-specified value while still maintaining an acceptable CBR in the presence of smoothly varying arrival rates of traffic, in which the state space is too large for practical deployment of the other learning scheme.

1 Introduction

Next Generation Wireless Systems are expected to support multimedia services with diverse quality of services (QoS), such as voice, video and data. Due to the rapid growth in mobile users and scarce radio resources, CAC has become vital to guarantee the QoS for the multiple services and utilize the network resources [Ahmed, 2005].

Generally a cellular network has a limited number of bandwidth units (BWU) or channels, which could be frequencies, time slots or codes depending on the radio access technique used, viz, FDMA, TDMA, or CDMA respectively. Arriving calls are accepted to or rejected from access to the network by the CAC scheme based on the predefined policy. As a user moves from one cell to another, the call requires reallocation of channels in the destination cell. This procedure is called handoff. If there are no available channels in the destination cell, the call may be prematurely terminated due to handoff failure, which is highly undesirable. The potential performance measures in cellular networks are long term revenue, utility, CBR (which is calculated by the number of rejected setup calls divided by the number of setup requests) or handoff failure rate (CDR, which is calculated by the number of rejected handoff calls divided by the number of handoff requests). CDR can be reduced by reserving some bandwidth for future handoffs. However, the CBR may increase due to such bandwidth reservation. Hence, reduction of CBR and CDR are conflicting requirements, and optimization of both is extremely complex.

Several researchers have shown that the guard channel threshold policy, which a priori reserves a set of channels in each cell to handle handoff requests, is optimal for minimizing the CDR in a non-multimedia situation. However the computational complexity of these approaches becomes too high in multi-class services with diverse characteristics context, and exact solutions become intractable [Soh and Kim, 2001]. [Choi, 2002] suggest a bandwidth reservation scheme using an effective mechanisms to predict a mobile terminal (MT)'s moving direction and reserve bandwidth dynamically based on the accurate handoff prediction. However this scheme incurs high communication overheads for accurate prediction of a MT's movement.

There are a variety of artificial intelligence techniques that have been applied to the CAC schemes, such as RL [Senouci, 2004]. However RL can be difficult to scale up to larger domains due to the exponential growth of the number of state variables. So in complex real world situations learning time is very long [Sutton, 1998].

This paper proposes a novel approach to solve the CAC in multimedia cellular networks with multiple classes of traffic with different resource and QoS requirements, and the traffic loads can vary according to the time. A near optimal CAC policy is obtained through a form of NE algorithm called NeuroEvolution of Augmenting Topologies (NEAT) [Stanley, 2004]. By prioritizing the handoff calls and perceiving the real time CBR and CDR, the proposed learning scheme called CN can be trained offline and the learned best policy can dynamically adjust the CAC policy to adapt to the varying of traffic loads: when the perceived CDR become higher, the system will reject more setup calls to decrease the CDR, and vice versa.

The learned policies are compared to a greedy scheme, which always accepts a call if there is enough bandwidth capacity to accept this call, and a RL CAC scheme, which is similar as the scheme proposed in [Senouci et al., 2004].The simulation results shows that our proposed scheme can learn very flexible and near optimal CAC policies, which can maintain the QoS constraints in the presence of smoothly changing arrival rate of traffic, a scenario for which the state space is too large for practical deployment of the other learning scheme evaluated.

The paper is organized as follows: Section 2 gives a brief introduction to NEAT, which is a form of NE Method, and describes how to apply the NEAT to the CAC application; section 3 describes the test system model, and formulates the fitness function, while section 4 compares the performance in constant and time-varying traffic loads.

2 Applying NEAT to CAC

NEAT is a kind of NE method that has been shown to work very efficiently in complex RL problems. NE is a combination of neural networks and genetic algorithms where neural networks (NNs) are the phenotype being evaluated. [Stanley, 2004]

The NEAT method for evolving artificial NNs is designed to take advantage of neural network structure as a way of minimizing the dimensionality of the search space. The evolution starts with a randomly generated small set of NNs with simple topologies. Each of these NNs is assigned a fitness value depending on how well it suits the solution. Once all the members of the population are assigned fitness values, a selection process is carried out where better individuals (high fitness value) stand a greater chance to be selected for the next operation. Selected individuals undergo recombination and mutation to result in new individuals. Structural mutations add new connections and nodes to networks in the population, leading to incremental growth. The whole process is repeated with this new population until some termination criterion is satisfied. [Stanley, 2004]

This section specifies the main requirements of applying NEAT to the CAC application, and describes the work process of setting up a connection.

2.1 Main Requirements

NEAT can be seen as a black box, which can provide a neural network for receiving the inputs and generating the outputs. There area several issues need to be considered:

1. What are the inputs?

Normally the inputs are the perceived state of the environment that is essential to make the action decision. For the CAC problem the perceived state may include the new request call and all the currently carried connections in the cell.

2. What are the outputs?

Generally the outputs are the possible actions that can be performed in the real application. In CAC, there are only two possible actions: Accept and Reject, so one output is enough. We define the output is a real number, and its value is between 0 and 1, if it is larger than 0.5, then the action selected is Accept; otherwise, it is Reject.

3. How to evaluate each policy and how to formulate the fitness function?

The fitness function determines what is good or bad policy. A good policy has a high fitness score and so will have a higher probability to create offspring policies. The fitness function is described in a later section.

4. Internal and External Supervisor

During the learning period, many NNs are randomly generated and some of them can lead to very bad performance. To prevent damage or unacceptable performance, an Internal Supervisor is needed to filter out clearly bad performance policies. In CAC, NNs that always reject all kinds of calls are frequently generated and so would be evaluated. However these Always Reject policies are obviously not good and evaluation is pointless. So the Internal Supervisor gives their fitness score the value of 0.

Additionally, the actions generated by NEAT are not always feasible. For example, during evaluation the evolved NNs may try to accept a request call when the system is full, which is physically unrealizable. Therefore an External Supervisor is added and uses prior knowledge in order to filtering impossible actions due to the system constraints.

2.2 The Work Process of Setting up a connection

Figure 1 illustrates the work process of setting up a connection using our scheme.

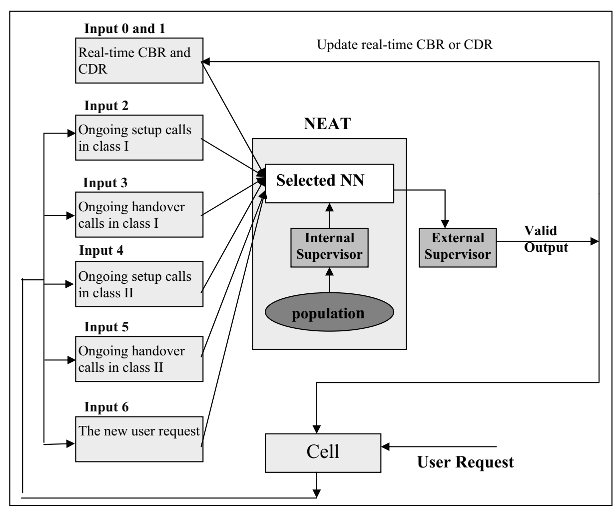

Figure 1 the process of setting up a connection

1. The network cell receives a user request event (in this case from a network traffic simulator) , perceives the network state (such as checking all the ongoing connections in the cell, calculates the real time CBR and CDR), and send the inputs to NEAT;

2. The NN currently being evaluated in NEAT generates the output according to the received inputs, and sends the output to the External Supervisor;
3. If the output is invalid as determined by the predefined rules in the External Supervisor, a Reject action will be sent to the cell. This occurs only when the capacity of cell is at its upper limit and the output decision of the CAC policy is still Accept. In this way the policy will not be punished by negative fitness value for creating impossible actions, because during exploration it is very difficult to create a policy that can always generate correct actions for any possible environment.
4. The cell performs the output action, which means to accept or reject the request event.
5. If the decision is to accept, the cell will allocate the requested bandwidth; if the decision is to reject, the cell will not take any action.

3 Fitness Function

Our objective of CAC is to accept or reject request (setup or hand off) calls so as to maximize the expected revenue over an infinite planning horizon and meet the QoS constraints, and more specifically to trade off CBR and CDR so that the specified CDR will not exceed a predefined upper bound while retaining an acceptable CBR level.

In this paper we only consider non-adaptive services, e.g. the bandwidth of a connection is constant. We assume m classes of traffic: $\{1, 2, ...m\}$. A connection in class i consumes b_i units of bandwidth or channels, and the network can obtain r_i rewards per logical second by carrying it. We also assume that traffic from each class arrives according to Poisson distribution and their holding time is exponentially distributed. All the arrival distributions and call holding distributions are independent of each other. The arrival events include new call arrival events and handoff arrival events. Since the call departures do not affect the CAC decisions, we only consider the arrival events in the CAC state space. Additionally, we only consider a cell with fixed capacity. The fixed total number of bandwidth (channels) is C.

We denote $\lambda_{i,s}$ and $\lambda_{i,h}$ as the arrival rates of new setup and handover requests in class i, $\mu_{i,s}^{-1}$ and $\mu_{i,h}^{-1}$ as the average holding time of setup and handover calls in class i.

3.1 First Goal: maximize revenue- F_1

If the goal is simply to maximize revenue, the fitness can be assessed by F_1, which can be calculated from the average reward obtained per request connection as equation (1). Let N be the total number of request connections (includes setup and handoff calls), n is the total number of accepted connections.

For large N, $$F_1 = \frac{\sum_{n=1}^{N} R_n}{N} \cong \frac{\sum_{i=0}^{m} r_i \left(k_{i,s}\mu_{i,s}^{-1} + k_{i,h}\mu_{i,h}^{-1}\right)}{N} \quad (1)$$

Here m denotes the total number of classes of traffic, i denotes the class of traffic for each individual connection, $k_{i,s}$ and $k_{i,h}$ denotes the number of the accepted new setup calls and handover calls in class i. The relationship below also holds.

$$k_{i,s} = N \frac{\lambda_{i,s}}{\sum_{i=0}^{m}(\lambda_{i,s} + \lambda_{i,h})} p_{i,s} \text{ and } k_{i,h} = N \frac{\lambda_{i,h}}{\sum_{i=0}^{m}(\lambda_{i,s} + \lambda_{i,h})} p_{i,h},$$

Where $p_{i,s}$ and $p_{i,h}$ denotes the acceptance rate of new setup request calls and handover calls in class i.

$$\begin{cases} p_{i,s} = \dfrac{\text{the number of accepted setup calls in class i}}{\text{request setup calls in class i}} \\ p_{i,h} = \dfrac{\text{the number of accepted handover calls in class i}}{\text{request handover calls in class i}} \end{cases}$$

To simplify the above formula, define the service demand parameter α as

$$\alpha_{i,s} = r_i \frac{\lambda_{i,s}}{\sum_{i=0}^{m}\left(\lambda_{i,s} + \lambda_{i,h}\right)} \mu_{i,s}^{-1} \text{ and } \alpha_{i,h} = r_i \frac{\lambda_{i,h}}{\sum_{i=0}^{m}\left(\lambda_{i,s} + \lambda_{i,h}\right)} \mu_{i,h}^{-1}$$

Therefore $$F_1 = \sum_{i=0}^{m}(\alpha_{i,s} p_{i,s} + \alpha_{i,h} p_{i,h}) \quad (2)$$

It can be seen that the total revenue is determined by both of the service demand parameters α and the learned CAC performance $p_{i,s}$ and $p_{i,h}$, which can be calculated after evaluation of each individual policy.

3.2 Handle CBR and CDR trade-off-- F_2.

Two parameters are used to trade off between CBR and CDR: the Importance Factor η and the Penalty Fitness f. The Importance Factor η indicates the preference of the system as shown in Equation (3). For a network it is commonly more desirable to reject new setup requests and accept more handoff requests, so normally $\eta_{i,s}\alpha_{i,s} < \eta_{i,h}\alpha_{i,h}$. By carefully selecting η for each kind of traffic, the expected CAC policy can be evolved. The fitness function is shown as below:

$$F_2 = \sum_{i=0}^{m}(\eta_{i,s}\alpha_{i,s}p_{i,s} + \eta_{i,h}\alpha_{i,h}p_{i,h}) - \sum_{i=0}^{m}(f_{i,s} + f_{i,h}) \quad (3)$$

The other parameters in Equation (3) are the Penalty Fitness f. When the observed CBR or CDR exceeds their predefined upper limit bound (T_{cdr} and T_{cbr}), the evaluated policy needs to be punished by a negative fitness score f, which should be large enough to affect the total fitness F_2. Additionally, although dropping handoff calls is usually more undesirable than blocking a new connection, designing a system with zero CDR is practically impossible and will lose unacceptable revenue. In practice, an acceptable CDR is around 1 or 2%. $f_{i,s}$ and $f_{i,h}$ give an opportunity to prevent the network from losing too much revenue by rejecting setup calls. Generally, we consider that CDRs are more important than CBRs, therefore $|f_{i,h}| > |f_{i,s}|$.

These two parameters η and f work cooperatively to trade off between CBRs and CDRs: η gives extra weight for each different kind of traffic to indicate the preference of the network, $f_{i,h}$ can increase the speed to find policies that can decrease CDR under T_{cdr} , and $f_{i,s}$ defines the upper limit of T_{cbr} to trade off between CBRs and CDRs.

4 Training Design and Performance Comparison

This section presents two kinds of training processes: one for constant traffic load, another for time-varying traffic load. We use computer simulations to evaluate the performance of the learning approach. A single cell within a cellular network with a limited number of channels $C = 20$ is considered. We define two classes of traffic labeled C_1 and C_2, which require bandwidth capacity 1 and 2 respectively. The reward rate is $r = 1$/sec/connection for carrying each connection.

To evaluate the traffic load in each cell, we choose to normalized offered load with respect to the radio bandwidth capacity in a wireless cell, which is defined as "Normalized offered load" mentioned in [Soh, 2001] .

$$L = \frac{1}{C}\left[\sum_{i=1}^{2}\left(\lambda_{i,s}\mu^{-1}{}_{i,s}b_i + \lambda_{i,h}\mu^{-1}{}_{i,h}b_i\right)\right]$$

The simulation is divided as two periods: a Learning Period and an Evaluation Period. During the Learning Period CN is trained to evolve higher fitness policies. During the evaluation period, the highest fitness policy is selected to perform CAC.

4.1 Constant traffic load

In this experiment the CN is compared with a Q Learning RL based scheme and a Greedy Policy, all for a constant normalized offered load. The traffic parameters is defined in Table 1.

Parameters	C_1		C_2	
	Setup	Handover	Setup	Handover
λ^{-1}	0.2	0.1	0.1	0.05
$\mu^{-1}{}_i$	30	20	20	15

Table 1.Traffic parameters

The proposed scheme--CN

Training in constant traffic loads, the service demand parameter α doesn't change, so the fitness is only concerned with the learned CAC policy $p_{i,s}$ and $p_{i,h}$. CN requires five inputs. Input 0 indicates the setup utility of different kinds of request event (the utility is calculated as equation(4)), and input 1 to 4 indicates the number of ongoing calls for each kinds of traffic carrying in the cell. For better performance these inputs are normalized from 0 to 1.

$$Utility = \frac{\text{(importance factor)}\,\eta \times \text{(reward rate)}\; r}{b \text{ (bandwidth capacity)}} \quad (4)$$

Equation (3) is used as the fitness function, and Table 2 shows the value of the related parameters. Comparing the values of $\eta_i\alpha_i$, the setup calls of class C_2 has lest value, therefore it has the most probability to be rejected.

Parameters	C_1		C_2	
	Setup	Handover	Setup	Handover
α_i	13.33	4.44	3.33	2.22
η_i	3	10	5	80
$\eta_i\alpha_i$	40	44.44	16.67	177.88

Table 2.CN simulation parameters

During learning period, NEAT evolves 13 generations, each generation had 200 policies, and each policy was evaluated for 20,000 events.

RL based CAC scheme

We implemented a lookup table based Q-Learning CAC scheme (RL-CAC), which is similar to the one proposed in [Senouci, 2004]. The Q-function is learned through the following update equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$$

The reward $r(s,a)$ assess the immediate payoff due to a action decision in time t that it is in state s and has taken action a . Since the network can obtains the rewards by carrying each connection in a while, and receive nothing when just accept or reject it, the immediate reward is calculated by the total rewards obtains during the time between two consecutive events: one is happened in time t , and another is the previous event.

- Learning rate $\alpha_t = \dfrac{1}{1 + visit_t(s_t, a_t)}$,

  where $visit_t(s_t, a_t)$ is the total number of times this state-action pair has been visited.
- Discount factor $0 \le \gamma \le 1$.
- Exploration probability $\varepsilon = 1$.
- Important factor η is used to differentiate each kind of traffic, and prioritize the handoff calls.

$\eta_{1,s}$	$\eta_{1,h}$	$\eta_{2,s}$	$\eta_{2,h}$
5	10	1	80

Table 3.Important Factor in RL-CAC

Greedy Policy

The system always accepts a request if there is enough bandwidth capacity to accept the new call.

Performance Comparison

Three CN schemes are compared with RL-CAC and the Greedy Policy. The total CBR and CDR used in Fig 2 evaluate the total rejected rate for setup and handoff calls.

As shown in Fig 2, the CN-C01, CN-C02 has similar performance with RL-CAC, which is evidence that the CN scheme obtains near optimal policy.

Table 4 and 5 shows that by defining different thresholds for each kinds of traffic, CN can learn very flexible CAC policies, and not only trade-off between setup and handoff calls but also trade-off between different classes of traffic, which is difficult implemented by RL based CAC Schemes. Because for such constraints semi-Markov decision problem (SMDP), the state space should be defined by QoS constraints, it becomes too large, and some form of function approximation is needed to solve the SMDP problem.

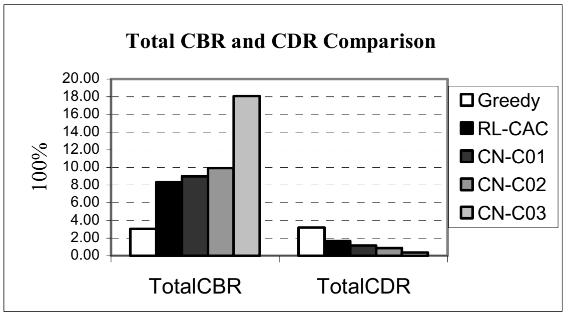

Figure 2. Total CBR and CDR Comparison

	T_{cbr}	Total CBR	T_{cdr}	Total CDR
CN-C02	10%	9.91%	1%	0.87%
CN-C03	20%	18.07%	0.50%	0.37%

Table 4.CN schemes comparison

CAC Schemes	Parameters	Total CDR	Total CBR	$C_{1,h}$	$C_{2,h}$	$C_{1,s}$	$C_{1,s}$
RL-CAC	CBR or CDR	1.68%	8.33%	1.1%	2.88%	3.49%	18.54%
CN-C01	Upper limit	No	No	1%	2%	10%	10%
	CBR or CDR	1.15%	8.97%	0.79%	1.89%	8.53%	9.91%

Table 5. CN and RL-CAC comparison

In addition, Simulation results show that by average CN schemes lost around 25% of revenue during learning by filtering clear bad performance policies.

4.2 Time-varying traffic load

In real world, the traffic load in a cellular system varies with time. A dynamic CAC policy which can adapt to the time-varying traffic loads is required: when the handoff failure rates grows higher, the system will reject more setup calls in order to carry more handoff requests, and vice versa. In this dynamic environment, most learning schemes trained in constant traffic loads cannot obtain the optimal performance and maintain the QoS constraints.

To simplify the problem, only the arrival rates of each kind of traffic are changed, the averages of holding times are kept as constant during the whole simulation.

In this dynamic environment, the state space is very large, and until now there is no technical report to provide any optimal solution by using RL algorithm. Therefore we only compare our schemes with Greedy Policy.

In the time-varying traffic loads, the service demand factor α varies according to the time and is difficult to be calculate, We use equation (5) to ignore the α, however it does not adapt to the changes of offered loads. In this case the CAC scheme requires feedback from the environment to indicate the changes of traffic loads. The simulation results demonstrate that the CBR and CDR can be good candidates. Because using any non-adaptive CAC policy, when the traffic load increases the CBR and CDR will become higher, the CBR and CDR can be a factor to reflect the traffic loads.

With time-varying traffic loads CN requires seven inputs (Figure 1): Input 0 indicates the real-time CBR calculated by the number of rejected setup calls divided by the latest 500 setup requests. If the number of setup requests is less than 500, then the number of rejected setup calls is divided by the number of setup request; Input 1 indicates the real-time CDR calculated in a similar way as input 0; Input 2 to 6 are same as the inputs in the experiment with constant traffic loads.

$$IF = 3p_{1,s} + 10p_{1,h} + 5p_{2,s} + 80p_{2,h} - \sum f \qquad (5)$$

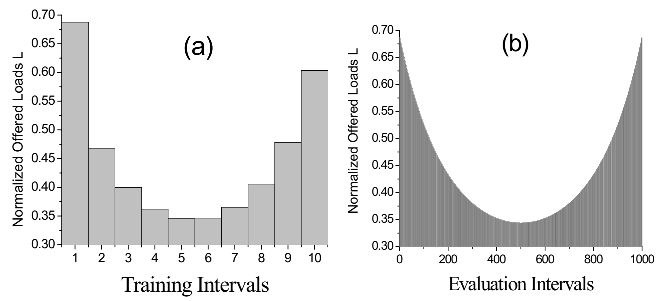

Figure 3 Traffic loads during learning and evaluation

The experiment is still divided into two periods: the Learning Period and Evaluation Period. In the Learning Period, each policy is trained with ten different normalized offered loads, which is called one Training Session (Figure 3.a). The load was generated by segmenting the Training Session into 10 time intervals where during each training interval the load was held at the constant level. The changes of λ_i^{-1} follow a sine function as shown in equation(6). Each policy is evaluated interval by interval and obtains an interval fitness score calculated by interval fitness function IF as equation (5). After finishing the evaluation of the 10th interval, the ten interval fitness scores are averaged as the final fitness of the policy.

$$\lambda_{learning}^{-1} = \lambda_{\min}^{-1} + \lambda_{\min}^{-1} \times \left|\sin(\frac{n\pi}{10} + \frac{\pi}{18})\right| \qquad (6)$$

$$\lambda_{evaluation}^{-1} = \lambda_{\min}^{-1} + \lambda_{\min}^{-1} \times \left|\sin(\frac{n\pi}{1000})\right| \qquad (7)$$

Parameters	C_1		C_2	
	Setup	Handover	Setup	Handover
$\lambda^{-1}{}_{\min}$	0.25	0.125	0.2	0.1
$\mu^{-1}{}_i$	25	20	10	5

Table 6.The traffic definition in time-varying traffic loads

The Evaluation Period evaluates the learned best policy. It is divided into 1000 equal lengths of time intervals with different values of arrival rates for each kind of traffic (Figure 3.b). The changes of λ^{-1} again follow a sine function as, which is similar as the one used in the learning phase, but sampled more finely. Within each evaluation interval the arrival rate is kept unchanged. Each evaluation interval is as long as one training interval. Table 4 shows the traffic definition.

The simulation runs 200,000,000 logical seconds. Each training or evaluation interval is 10,000 logical seconds with around 20,000 request events. NEAT evolves 20 generations, and each generation has 100 policies.

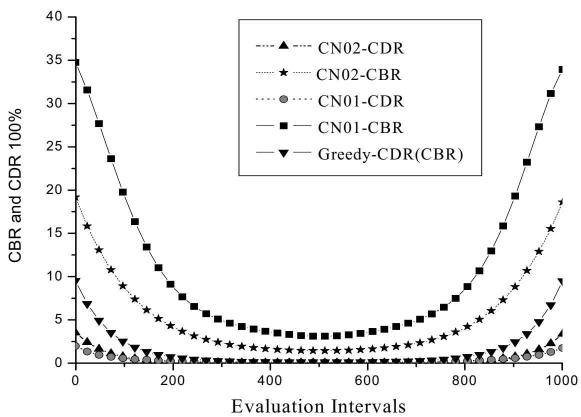

Figure 4.Comparison of interval CBR and CDR with time-varying traffic loads. At the end of each evaluation interval, the total CBR and CDR is calculated and compared.

Two CN schemes are compared: $CN01$ only specified the threshold for CDR, which is 2%; $CN02$ specified both of the CDR (2%) and CBR (varying according to the different training intervals).

Evaluation Period	Greedy	CN01	CN02
Total CBR	1.39	5.11	7.61
Total CDR	1.44	0.33	0.51

Table 7.Comparison of CBR and CDR

Figure 4 shows the comparison of CBR and CDR for different CAC schemes. We can see that with time-varying traffic loads CN can successfully maintains the CDR under a predefined level. Additionally from Table 7, it can be seen that the CN01 only defines the threshold for CDR therefore its CBR is higher, the system will lost more revenue by rejecting too many setup calls, however CN02 uses threshold for both of the CBR and CDR to force the system to decrease the CBR so as to successfully trade-off between the CBR and CDR.

In this experiment, the highest fitness NNs evolved by NEAT are very complex and non-standard, and it is impossible to specify any particular hidden layers, there are sets of hidden nodes connected with each other by negative or positive weights.

5 Conclusion

This paper proposes a novel learning approach to solve the CAC in multimedia cellular networks with multiple classes of traffic. The near optimal CAC policy is obtained through a form of NeuroEvolution (NE) algorithm.

Simulation results show that the new scheme can effectively guarantee the specified CDRs remain under their predefined upper bound while retaining acceptable CBRs. Additionally our proposed scheme can learn very flexible CAC policies by defining different thresholds for each kind of traffic, whereas RL based CAC scheme can not.

Furthermore this scheme can learn dynamic CAC policies to adapt to changes of traffic loads. No matter what the traffic load is, the CAC scheme is trained to reject more setup calls to decrease the CDR when the CDR exceeds the prescribed upper limit, and vice versa. Therefore if the scheme is trained carefully according to the traffic instance during a day in a network element, the highest fitness policy can be used directly in the real world.

Acknowledgments

Thanks Kenneth O. Stanley and Ugo Vieruccifor very much for providing free software of NEAT in the website http://www.cs.utexas.edu/~nn/index.php.

References

[Ahmed, 2005] Mohaned Hossam Ahmed. Call Admission Control in Wireless Networks: A Comprehensive Survey. Memorial University of Newfoundland, *IEEE Communications Surveys&Tutorials*,2005.

[Choi et al., 2001] Sunghyun Choi, and Kang G. Shin. Adaptive bandwidth reservation and admission control in QoS-sensitive cellular networks. In *IEEE Transactions on parallel and distributed systems,* Vol. 13, September 2002.

[Ho et al., 1999] Chi-Jui Ho and Chin-Tau Lea. Improving call admission policies in wireless networks. *Wireless Networks*, 5(1999):257–265, 1999.

[Senouci et al., 2004] S.M. Senouci, A.Beylot, G. Pujolle. Call Admission Control in Cellular Networks: A Reinforcement Learning Solution. *International journal of network management,* page 89-103, 2004.

[Soh and Kim, 2001] Wee-Seng Soh and Hyong S. Kim. Dynamic guard bandwidth scheme for wireless broadband networks". *IEEE INFCOM vol.1,* pages 572-581,2001. 0-7803-7016-3/01.

[Stanley, 2004] Kenneth O. Stanley. Efficient Evolution of Neural Networks through Complexification. *PhD thesis*. Artificial Intelligence Laboratory, the University of Texas at Austin, TX 78712, 2004

[Sutton and Barto, 1998] Richard S. Sutton, Andrew G. Barto. Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, 1998