7

Efficient Algorithm to Determine the Optimal Configuration of Admission Control Policies in Multiservice Mobile Wireless Networks

David Garcia-Roger, Jorge Martinez-Bauset and Vicent Pla

Departamento de Comunicaciones, Universidad Politécnica de Valencia, UPV, ETSIT, Camino de Vera s/n, 46022 Valencia, Spain; e-mail: dagarro@doctor.upv.es, {jmartinez, vpla}@dcom.upv.es

Abstract

We propose a new methodology and associated algorithms for computing the optimal configuration of the Multiple Fractional Guard Channel (MFGC) admission control policy in multiservice mobile wireless networks. Our approach is based on the solution space concept which discloses a novel insight into the problem of determining the optimal configuration parameter values of the MFGC policy and provides an heuristic evidence that the algorithm finds the optimal solution and converges in all scenarios, an evidence that was not provided in previous proposals. Besides, our algorithm is shown to be more efficient than previous algorithms appeared in the literature.

Keywords: Gradient methods, land mobile radio cellular systems, Markov processes, modeling, multimedia systems, optimal control.

7.1 Introduction

The enormous growth of mobile telecommunication services, together with the scarcity of radio spectrum has led to a reduction of the cell size in cellular

D. Kouvatsos (ed.), Mobility Management and Quality-of-Service for Heterogeneous Networks, 1–18.

systems. Smaller cell size entails a higher handover rate having an important impact on the radio resource management and the QoS perceived by customers. Moreover, 3G networks establish a new paradigm with a variety of services having different QoS needs and traffic characteristics. In these scenarios Admission Control (AC) is a key aspect in the design and operation of multiservice mobile networks.

In this paper we propose a new algorithm for computing the optimal configuration of a trunk reservation policy named the *Multiple Fractional Guard Channel* (MFGC) [1, 2]. The configuration of the MFGC policy specifies the average amount of resources that each service has access to. The optimal configuration maximizes the offered session rate that the system can handle while meeting certain QoS requirements, which we call the system capacity. The QoS requirements are defined as upper bounds for the blocking probabilities of both new setup and handover requests. In a wireless scenario this distinction is required because a session being forced to terminate due to a handover failure is considered more harmful than the rejection of a new session setup request. One of the important features of the MFCG policy is that it can achieve a system capacity that is very close to the optimal [3].

To the best of our knowledge only two algorithms for computing the system capacity of the MFGC policy have been proposed in the literature [2, 4]. We refer to those algorithms as HCO and PMC respectively, after their authors' initials. Our work is motivated by the fact that previous algorithms did not provide any evidence supporting that they where finding the optimal solution nor that they converged in all scenarios. Our approach provides a novel insight into the problem, which we believe that by itself it is a significant contribution, but in addition the algorithm we have developed, based on the insight provided by our study, offers computational advantages better than those provided by previous proposals.

The HCO algorithm requires the optimal *prioritization order* as input, i.e. a list of session types sorted by their relative priorities. For a system with N services, new session and handover request arrivals are considered, making a total of $2N$ arrival streams. Therefore, the MFGC policy configuration is defined by the $2N$-tuple $\boldsymbol{t} = (t_1, \ldots, t_{2N})$, where the configuration parameter $t_i \in \mathbb{R}$ represents the average amount of resources that stream i can dispose of. If $\boldsymbol{t}_{\text{opt}}$ is the policy setting for which the maximum capacity is achieved, the optimal prioritization order is the permutation $\boldsymbol{\sigma}^* \in \boldsymbol{\Sigma}$, $\boldsymbol{\Sigma} := \{(\sigma_i, \ldots, \sigma_{2N}) : \sigma_i \in \mathbb{N}, 1 \leq \sigma_i \leq 2N\}$, such that $t(\sigma_1^*) \leq t(\sigma_2^*) \leq \ldots \leq t(\sigma_{2N}^*) = C$, where $t(\sigma_i^*)$ is the σ_i^* element of $\boldsymbol{t}_{\text{opt}}$ and C is the total number of resource units of the system. Selecting the optimal prioritization order is a complicated task as it depends on both QoS constraints and system characteristics as pointed out in [2]. In general there are a total of $(2N)!$ different prioritization orders. In [2] the authors give some guidelines to construct a partially sorted list of prioritization orders according to their likelihood of being the optimal ones. Then a trial and error process is followed using successive elements of the list until the optimal prioritization order is found. For each element the HCO algorithm is run and if after a large number of iterations it did not converged, another prioritization order is tried.

The PMC algorithm does not require any a priori knowledge. Indeed, after obtaining the optimal policy configuration $\boldsymbol{t}_{\mathrm{opt}}$ for which the maximum capacity is achieved, the optimal prioritization order is automatically determined as a by-product of the algorithm. Moreover, through numerical examples it is shown in [4] that the PMC algorithm is more efficient than the HCO algorithm even when the latter is provided with the optimal prioritization order. In [4] the optimization problem is formulated as a non-linear programming problem, which attempts to determine the MFGC policy configuration parameters in such a way to maximize the session arrival rates while keeping the blocking probabilities under specified bounds, and an algorithm for solving the non-linear programming problem is provided. Given that, in general, the blocking probabilities are non-monotonic functions both of the offered load and the thresholds that specify the policy configuration, finding the optimal solution is not an easy task and no evidence was provided supporting that the algorithm converged in all scenarios.

Our new algorithm is based on the *solution space* concept. If for each possible configuration of the MFGC policy we determine the maximum session rate that can be offered to the system while satisfying the QoS constrains, then the result of this study is called the solution space. As with the HCO and PMC algorithms, the convergence of our algorithm is based on the assumption that the solution space has a single (and thus, global) maximum. Even though a formal verification of this assumption is out of the scope of the paper, we have obtained the solution space for multiple policies and multiple scenarios and found that a single peak can always be found in the solution space, and that this peak is the system capacity [3]. Besides, the shape of the solution spaces tend to be steeper for policies that achieve higher system capacity, like the MFGC policy. These evidences suggest that a simple *hill climbing* algorithm could be deployed, and might shed light on a more formal characterization of solution spaces.

The remaining of the paper is structured as follows. In Section 7.2 the system model is described and its mathematical analysis is outlined in Section 7.3. Section 7.4 justifies the applicability of a gradient method for the determination of the optimal configuration of the MFGC policy. Section 7.5 describes in detail the new proposed algorithm. Computational complexity of the algorithm is comparatively evaluated in Section 7.6. Finally, Section 7.7 concludes the paper.

7.2 Model Description

The system has a total of C resource units, being the physical meaning of a unit of resource dependent on the specific technological implementation of the radio interface. The system offers N different classes of service. For each service new and handover session request arrivals are distinguished so that there are N types of services and $2N$ types of arrival streams. Arrivals are numbered in such a manner that for service i new session arrivals are referred to as arrival type i, whereas handover arrivals are referred to as arrival type $N+i$.

For the sake of mathematical tractability we make the common assumptions of Poisson arrival processes and exponentially distributed random variables for cell residence time and session duration.

The arrival rate for new (handover) sessions of service i is λ_i^n (λ_i^h). A request of service i consumes b_i resource units, $b_i \in \mathbb{N}$. We denote by f_i the percentage of service i new session requests and assume that its value is known. Therefore, the aggregated rate of new session requests is expressed as $\lambda^T = \sum_{i=1}^{N} \lambda_i^n$, $\lambda_i^n = f_i \lambda^T$. This is a common simplification in the literature [5].

The duration of service i sessions is exponentially distributed with rate μ_i^c. The cell residence time of a service i customer is exponentially distributed with rate μ_i^r. Hence, the resource holding time in a cell for service i is exponentially distributed with rate $\mu_i = \mu_i^c + \mu_i^r$. The exponential assumption for the cell residence time represents a good performance approximation and indicates general performance trends [6]. The exponential assumption can also be considered a good approximation for the time in the handover area [7] and for the interarrival time of handover requests [8].

Let $\boldsymbol{p} = (P_1, \ldots, P_{2N})$ be the blocking probabilities, where new session blocking probabilities are $P_i^n = P_i$ and the handover ones are $P_i^h = P_{N+i}$. The forced termination probability of accepted sessions under the assumption of homogeneous cell [9] is

$$P_i^{ft} = \frac{P_i^h}{\mu_i^c/\mu_i^r + P_i^h}.$$

The system state is described by an N-tuple $\boldsymbol{x} = (x_1, \ldots, x_N)$, where x_i represents the number of type i sessions in the system, regardless they were initiated as new or handover sessions. This approximation is irrelevant when considering exponential distributions due to their memoryless property. Let $b(\boldsymbol{x})$ represent the amount of occupied resources at state $\boldsymbol{x}$, $b(\boldsymbol{x}) = \sum_{i=1}^{N} x_i b_i$.

A generic definition of the MFGC and Complete-Sharing policies are now provided. For the MFGC policy, when a service i request finds the system in state $\boldsymbol{x}$, the following decisions can be taken:

$$b(\boldsymbol{x}) + b_i \begin{cases} \leq \lfloor t_i \rfloor & \text{accept} \\ = \lfloor t_i \rfloor + 1 & \text{accept with probability} \quad t_i - \lfloor t_i \rfloor \\ > \lfloor t_i \rfloor + 1 & \text{reject} \end{cases}$$

where parameters t_i are the policy configuration parameters that are set to achieve a given QoS objective.

The Complete-Sharing (CS) policy is equivalent to the absence of policy, i.e. a request is admitted provided there are enough free resource units available in the system.

7.3 Mathematical Analysis

The model of the system is a multidimensional birth-and-death process, which state space is denoted by S. Let $r_{\boldsymbol{xy}}$ be the transition rate from $\boldsymbol{x}$ to $\boldsymbol{y}$ and let $\boldsymbol{e}_i$ denote a vector whose entries are all 0 except the i-th one, which is 1.

$$r_{\boldsymbol{xy}} = \begin{cases} a_i^n(\boldsymbol{x})\lambda_i^n + a_i^h(\boldsymbol{x})\lambda_i^h & \text{if } \boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e}_i \\ x_i \mu_i & \text{if } \boldsymbol{y} = \boldsymbol{x} - \boldsymbol{e}_i \\ 0 & \text{otherwise} \end{cases}$$

The coefficients $a_i^n(\boldsymbol{x})$ and $a_i^h(\boldsymbol{x})$ denote the probabilities of accepting a new and handover session of service i respectively. Given a policy configuration

$(t_1, \ldots, t_{2N})$ these coefficients can be determined as follows

$$a_i^n(\boldsymbol{x}) = \begin{cases} 1 & \text{if } b(\boldsymbol{x}) + b_i \leq \lfloor t_i \rfloor \\ t_i - \lfloor t_i \rfloor & \text{if } b(\boldsymbol{x}) + b_i = \lfloor t_i \rfloor + 1 \\ 0 & \text{if } b(\boldsymbol{x}) + b_i > \lfloor t_i \rfloor + 1 \end{cases}$$

and

$$a_i^h(\boldsymbol{x}) = \begin{cases} 1 & \text{if } b(\boldsymbol{x}) + b_i \leq \lfloor t_i \rfloor \\ t_{N+i} - \lfloor t_{N+i} \rfloor & \text{if } b(\boldsymbol{x}) + b_i = \lfloor t_{N+i} \rfloor + 1 \\ 0 & \text{if } b(\boldsymbol{x}) + b_i > \lfloor t_{N+i} \rfloor + 1 \end{cases}$$

From the above, the global balance equations can be written as

$$p(\boldsymbol{x}) \sum_{\boldsymbol{y} \in S} r_{\boldsymbol{x}\boldsymbol{y}} = \sum_{\boldsymbol{y} \in S} r_{\boldsymbol{y}\boldsymbol{x}} \, p(\boldsymbol{y}) \qquad \forall \boldsymbol{x} \in S \tag{7.1}$$

where $p(\boldsymbol{x})$ is the state $\boldsymbol{x}$ stationary probability. The values of $p(\boldsymbol{x})$ are obtained from (7.1) and the normalization equation. From the values of $p(\boldsymbol{x})$ the blocking probabilities are obtained as

$$P_i = P_i^n = \sum_{\boldsymbol{x} \in S} \big(1 - a_i^n(\boldsymbol{x})\big) p(\boldsymbol{x}) \qquad P_{N+i} = P_i^h = \sum_{\boldsymbol{x} \in S} \big(1 - a_i^h(\boldsymbol{x})\big) p(\boldsymbol{x})$$

If the system is in statistical equilibrium the handover arrival rates are related to the new session arrival rates and the blocking probabilities (P_i) through the expression [10]

$$\lambda_i^h = \lambda_i^n \frac{1 - P_i^n}{\mu_i^c / \mu_i^r + P_i^h} \tag{7.2}$$

The blocking probabilities do in turn depend on the handover arrival rates yielding a system of non-linear equations which can be solved using a fixed point iteration method as described in [9, 10].

7.4 Determination of the Optimal Policy Configuration

We pursue the goal of computing the system capacity, i.e. the maximum offered session rate that the network can handle while meeting certain QoS requirements. These QoS requirements are given in terms of upper-bounds for the new session blocking probabilities (B_i^n) and the forced termination probabilities (B_i^{ft}). The common approach to carry out this AC synthesis process in multiservice systems is by iteratively executing an analysis process. The synthesis process is a routine that having as inputs the values of the system parameters (λ_i^n, λ_i^h, μ_i, b_i and C) and the QoS requirements (B_i^n and B_i^{ft}), produces as output the optimal configuration (the thresholds t_i). In contrast the analysis process is a routine that having as inputs the value of the system parameters and the configuration of the AC policy produces as output the blocking probabilities for the different arrival streams.

Given that, in general, the blocking probabilities are non-monotonic functions both of the offered load and the thresholds that specify most policy configurations; the common approach is to carry out a multidimensional search using, for example, meta-heuristics like genetic algorithms which are able to find a *good* configuration in a reasonable amount of time. It should be pointed out that each execution of the analysis process requires solving the associated continuous-time Markov chain.

Additional insight can be gained by determining the maximum offered session rate for each possible policy configuration. The result of this study is called the *solution space*, and its peak value is the system capacity of the AC policy, i.e. the maximum aggregated session arrival rate ($\lambda^T = \sum_{i=1}^N \lambda_i^n$, $\lambda_i^n = f_i \lambda^T$) that can be offered to the system in order to satisfy the QoS requirements. The surface that defines the solution space is obtained as follows. For each configuration of the thresholds t_i, $\lambda_{\max}^T$ is computed by a binary search process which has as input the value of the system parameters μ_i, b_i, C and the thresholds t_i, and produces as output the blocking probabilities (P_i^n and P_i^h). The binary search process stops when it finds the $\lambda_{\max}^T$ that meets the QoS requirements (B_i^n and B_i^{ft}), $i = 1, \ldots, N$.

In order to illustrate our algorithm we have chosen a simple example with only two services but without their associated handover streams. This allows us to represent the solution space in only three dimensions. Figure 7.1 show the solution space when MFGC policy is deployed in a scenario with $C = 10$ resource units, $\boldsymbol{b} = (1, 2)$, $\boldsymbol{f} = (0.8, 0.2)$, $\boldsymbol{\mu} = (1, 3)$, $\boldsymbol{B}^n = (0.05, 0.01)$. The configuration of the policy is defined by two parameters t_1 and t_2. It should be noted that the system capacity is expressed as a relative value to the capacity obtained for the CS policy.

The form of the solution space shown in Figure 7.1, which displays a unique maximum, suggests that a hill climbing algorithm could be an efficient approach to obtain the optimum configuration for MFGC policy in this scenario. Other optimization approaches like genetic algorithms that are more appropriate for scenarios with multiple local maxima will not be as efficient. We have obtained the solution space for multiple policies and multiple scenarios and found that a single peak can always be found in the solution space, and that this peak is the system capacity [3]. The policies that meet this condition are defined below. The solution space has been obtained for the scenarios defined in Table 7.1. To simplify the description of the admission policies we define the vector $\boldsymbol{x}' = (x_1^n, \ldots, x_N^n, x_1^h, \ldots, x_N^h)$, where x_i^n (x_i^h) are the sessions in progress of service i initiated as new (handover). It is clear that $x_i = x_i^n + x_i^h$.

Integer Limit (IL) [11]. A threshold (t; $t \in \mathbb{N}$, $1 \leq t \leq C$) is defined for each request type. Thus, there will be a set of $2N$ thresholds $(t_1^n, \ldots, t_N^n, t_1^h, \ldots, t_N^h)$. A request of type x_i^n (x_i^h) that arrives in state $\boldsymbol{x}'$ is accepted if $x_i^n + 1 \leq t_i^n$ ($x_i^h + 1 \leq t_i^h$) and blocked otherwise.

Upper Limit and Guaranteed Minimum (ULGM) [12]. Service i requests have access to two sets of resources: a private set and a shared set. The number of resource units in the private set available for new setup requests is denoted as $s_i^n \cdot b_i$ and for handover requests as $s_i^h \cdot b_i$, where $s_i^n, s_i^h \in \mathbb{N}$. Therefore the size of the shared set is $C - \sum_{i=1}^{N}(s_i^n + s_i^h) \cdot b_i$. A new (handover) service i request is accepted if $(x_i^n + 1) \leq s_i^n$ ($(x_i^h + 1) \leq s_i^h$) or if there are enough free resource units in the shared set, otherwise it is blocked.

Multiple Guard Channel (MGC) [13]. A threshold (t; $t \in \mathbb{N}$, $1 \leq t \leq C$) is defined for each request type. Thus, there will be a set of $2N$ thresholds $(t_1^n, \ldots, t_N^n, t_1^h, \ldots, t_N^h)$. A request of type x_i^n (x_i^h) that arrives in state $\boldsymbol{x}$ is accepted if the number of busy resource units is less than the corresponding threshold, i.e. $b(\boldsymbol{x}) + b_i \leq t_i^n$ ($b(\boldsymbol{x}) + b_i \leq t_i^h$), and blocked otherwise.

Multiple Fractional Guard Channel (MFGC). It was defined in Section 7.2.

Figure 7.1 shows how the hill climbing algorithm works. (i) Given a starting point in a $2N$-dimensional search space (for example, point **0**), the hill climbing algorithm begins by computing the value of the function (the system capacity $\lambda_{\max}^T$), and the blocking probabilities for the different arrival streams (P_i^n and P_i^h); (ii) the steepest dimension is selected as described below (in this case t_2^n); (iii) the algorithm searches for a maximum point along that dimension (in this case **1**). Note that what we have here is actually a maximization problem along a line; and (iv) return to (i) until the local

Table 7.1 Definition of the scenarios studied by Garcia et al. [3].

	b_1	b_2	f_1	f_2	$B_1^n(\%)$	$B_2^n(\%)$	$B_i^h(\%)$	λ_i^n	λ_i^h	μ_1	μ_2
A	1	2	0.8	0.2	5	1					
B	1	4	0.8	0.2	5	1					
C	1	2	0.2	0.8	5	1	$0.1B_i^n$	$f_i\lambda$	$0.5\lambda_i^n$	1	3
D	1	2	0.8	0.2	1	2					
E	1	2	0.8	0.2	1	1					

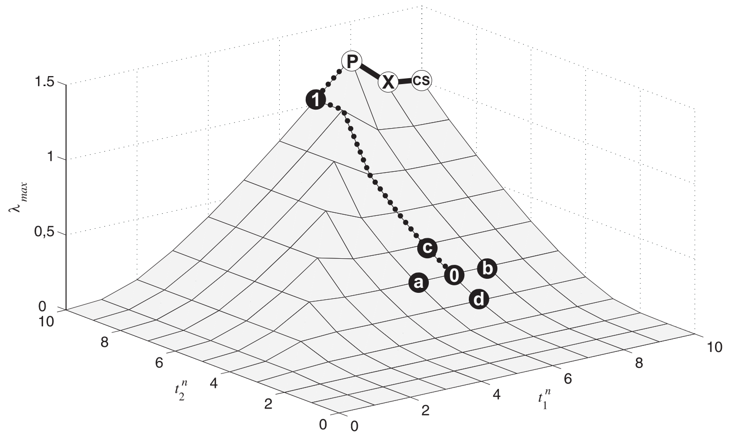

Figure 7.1 Example of the use of a hill climbing algorithm to determine the optimum configuration for the MFGC policy.

maximum **P** is found within the desired precision (the algorithm progression is shown with a dotted line). For the hill climbing algorithm explained, two important questions that arise are: (a) how the steepest dimension is selected, and (b) how the search for a maximum point is performed.

When applying gradient based methods to our problem, the function must be evaluated for the two adjacent neighbors in each of the 2N dimensions (points **a**, **b**, **c** and **d**), selecting then the steepest dimension as the one for which the function value is largest (**c**). However, if the binary search process to compute the system capacity is performed with low accuracy, or neighbors are sufficiently close to the considered point, then this method is impracticable, mainly because the function values for the neighbors are identical to the considered point, giving no information at all.

As this is the case, another approach is required to determine the steepest dimension. We define the relative distance to the QoS objective B_i of an arrival stream i with blocking probability P_i (assuming that meets its objective, i.e. $P_i < B_i$) by the quotient $(B_i - P_i)/B_i$. Since the relative distance to the QoS objective of an arrival stream is usually an indication that requests from that arrival stream may be blocked further while still meeting the QoS objective (and thus, providing an additional capacity for the remaining arrival streams), we choose as the steepest dimension the configuration parameter t_i associated with the arrival stream i which maximizes the relative distance to the QoS objective. Besides, this approach has an additional benefit in terms of computational complexity: it removes the need for computing the system capacity in the neighboring points in order to find out the steepest dimension.

In its search for the maximum along this steepest dimension our algorithm makes a number of successive unitary steps and it stops when it reaches the peak. When the solution space is continuous, as with the MFGC policy, a gradual refinement process is needed to reduce the size of the step once a promising region has been found, which is possibly close to the optimal. A further reduction of the computation complexity can be obtained by observing that the optimum configuration (point **P**) for any policy is near the CS configuration (point **CS**), and therefore it is a good idea to select it as the starting point. Figure 7.1 illustrates a typical progression (solid line) of the proposed algorithm starting from the CS configuration (point **CS**) towards point **X** and ending at the peak (point **P**).

7.5 Hill Climbing Algorithm

The capacity optimization problem can be formally stated as follows

Given: $C, b_i, f_i, \mu_i^c, \mu_i^r, B_i^n, B_i^{ft}\ ; i = 1, \ldots, N$

Maximize: $\lambda^T = \sum_{1 \le i \le N} \lambda_i^n, \lambda_i^n = f_i \lambda^T$
by finding the appropriate MFGC parameters t_i; $i = 1, \ldots, 2N$

Subject to: $P_i^n \le B_i^n, P_i^{ft} \le B_i^{ft}\ ; i = 1, \ldots, N$

We propose an algorithm to work out this capacity optimization problem. Our algorithm has a main part (Algorithm 1 `solveMFGC`) from which the procedure `capacity` (see Algorithm 2) is called. The procedure `capacity` does, in turn, call another procedure (`MFGC`) that calculates the blocking probabilities. For the sake of notation simplicity we introduce the 2N-tuple

$\boldsymbol{p}_{\mathbf{max}} = (B_1^n, \ldots, B_N^n, B_1^h, \ldots, B_N^h)$ as the upper-bounds vector for the blocking probabilities, where the value of B_i^h is given by

$$B_i^h = \frac{\mu_i^c}{\mu_i^r} \frac{B_i^{ft}}{1 - B_i^{ft}} \tag{7.3}$$

Following the common convention we used bold-faced font to represent array variables in the pseudo-code of algorithms.

Algorithm 1 $(\lambda^T_{\max},\ \boldsymbol{t}_{\text{opt}})$=`solveMFGC`$(C, \boldsymbol{p}_{\mathbf{max}}, \boldsymbol{b}, \boldsymbol{\mu_c}, \boldsymbol{\mu_r})$ (calculates MFGC parameters)

ε_2 :=< desired precision >
currentε_2 := 1
point := C
direction := -1
$\boldsymbol{step}$:= $(1, 1, \ldots, 1)$ <size $2N$>
steepest := 0
changeOfDirection:=FALSE
$\boldsymbol{t}_{\text{opt}} := (C, C, \ldots, C)$; $\boldsymbol{t} := \boldsymbol{t}_{\text{opt}}$
$\lambda^T_{\max} := \mathbf{0}$; $\lambda^T := \mathbf{0}$;
$\boldsymbol{p}_{\text{opt}} := \mathbf{0}$; $\boldsymbol{p} := \mathbf{0}$;
$\boldsymbol{dp}_{\text{opt}} := \mathbf{0}$; $\boldsymbol{dp} := \mathbf{0}$;

$(\lambda^T_{\max}, \boldsymbol{p}_{\text{opt}})$:=capacity$(\boldsymbol{p}_{\mathbf{max}}, \boldsymbol{t}_{\text{opt}}, \boldsymbol{\mu_c}, \boldsymbol{\mu_r}, \boldsymbol{b}, C)$
$\boldsymbol{dp}_{\text{opt}} := (\boldsymbol{p}_{\mathbf{max}} - \boldsymbol{p}_{\text{opt}})/\boldsymbol{p}_{\mathbf{max}}$; $\boldsymbol{dp} := \boldsymbol{dp}_{\text{opt}}$
currentε_2 := $max(\boldsymbol{dp}_{\text{opt}})$
steepest := < the stream i which maximizes $\boldsymbol{dp}_{\text{opt}}(i)$ >

while current$\varepsilon_2 > \varepsilon_2$ **do**
  point:= $\boldsymbol{t}_{\text{opt}}$(steepest)
  direction:= -1
  if $\boldsymbol{step}$(steepest)<>1 **then**
    $\boldsymbol{step}$(steepest)=0.5
  end if
  changeOfDirection:= FALSE

  repeat
    if direction= -1 **then**
      point=point$-\boldsymbol{step}$(steepest)
    else

```
            point=point+step(steepest)
        end if

        t := t_opt; t(steepest):= point;
        (λ^T,p) := capacity(p_max, t, μ_c, μ_r, b, C)
        dp := (p_max − p)/p_max;

        if λ^T >= λ^T_max then
            t_opt(steepest):= point; λ^T_max := λ^T;
            p_opt := p; dp_opt := dp;
        end if

        if dp(steepest)> ε_2 then
            if direction= −1 then
                if changeOfDirection then
                    step(steepest) := step(steepest)/2
                end if
            else
                step(steepest) := step(steepest)/2
                direction:= −1
                changeOfDirection:= TRUE
            end if
        else
            if λ^T < λ^T_max then
                if direction:= +1 then
                    if changeOfDirection then
                        step(steepest) := step(steepest)/2
                    end if
                else
                    step(steepest) := step(steepest)/2
                    direction:= +1
                    changeOfDirection:= TRUE
                end if
            end if
        end if
    until (dp(steepest)< ε_2) AND (λ^T >= λ^T_max)

    steepest:= < the stream i which maximizes dp_opt(i) >
    currentε_2 := max(dp_opt)
end while
```

Algorithm 2 $(\lambda^T_{\max}, \boldsymbol{p})$=`capacity`$(\boldsymbol{p}_{\mathbf{max}}, \boldsymbol{t}, \boldsymbol{\mu_c}, \boldsymbol{\mu_r}, \boldsymbol{b}, C)$

INPUTS: $\boldsymbol{p}_{\mathbf{max}}, \boldsymbol{t}, \boldsymbol{\mu_c}, \boldsymbol{\mu_r}, \boldsymbol{b}, C$
OUTPUTS: $\lambda^T_{\max}, \boldsymbol{p}$

1: $\varepsilon_1 :=<$ desired precision $>$
2: current$\varepsilon_1 := 1$
3: $L := 0$
4: $U :=<$ high value $>$
5: meetQoSrequirements:=FALSE
6:
7: **while** (current$\varepsilon_1 > \varepsilon_1$) OR NOT(meetQoSrequirements) **do**
8:   $\lambda^T_{\max} := (U + L)/2$
9:   $\boldsymbol{p} :=$ MFGC$(\boldsymbol{t}, \boldsymbol{\lambda_n}, \boldsymbol{\mu_c}, \boldsymbol{\mu_r}, \boldsymbol{b}, C)$
10:   current$\varepsilon_1 := min((\boldsymbol{p}_{\mathbf{max}} - \boldsymbol{p})/\boldsymbol{p}_{\mathbf{max}})$
11:   **if** current$\varepsilon_1 < 0$ **then**
12:     $U := \lambda^T_{\max}$
13:     meetQoSrequirements:= FALSE
14:   **else**
15:     $L := \lambda^T_{\max}$
16:     meetQoSrequirements:= TRUE
17:   **end if**
18: **end while**

The algorithm `solveMFGC` begins computing the CS configuration as the starting point (lines 13–14), and selects the arrival stream i for which $(B_i - P_i)/B_i$ is the highest as the steepest dimension, (line 16). The whole hill climbing loop starts in line 18, and the maximization loop along a dimension starts in line 26. Note (line 21) that the first time a dimension is chosen as the steepest, the hill climbing step equals to 1, however if a dimension has been chosen previously, initial hill climbing steps will be reduced to 0.5 because a certain locality of the optimal configuration is assumed. Lines (37–64) perform the hill climbing algorithm along the steepest dimension. This subroutine performs tasks like changing the direction of the successive steps and its refinement once a promising configuration has been found out. The algorithm `capacity` is basically a binary search of $\lambda^T_{\max}$ that calls procedure (`MFGC`) in each iteration in order to calculate the blocking probabilities.

7.5.1 On the Procedure `MFGC`

The procedure `MFGC`, which is invoked in the inner-most loop of our algorithm, is used to obtain the blocking probabilities, $\boldsymbol{p} := \mathrm{MFGC}(\boldsymbol{t}, \boldsymbol{\lambda_n}, \boldsymbol{\mu_c}, \boldsymbol{\mu_r}, \boldsymbol{b}, C)$, by using the Gauss–Seidel method to solve the continuous-time Markov chain (CTMC) that models the system. The major part of the computational complexity of the algorithms described in this paper comes from solving many different times the CTMC, therefore the difference among different algorithms basically yields on how many times a CTMC has to be solved. Note that in Section 7.6 only two types of services will be considered. For scenarios with a higher number of services, the Markov chain would have $2N$ dimensions (new and handover requests). To obtain valuable results the number of resource units of the system would have to be dimensioned appropriately, which will cause an explosion of the state space and thus rendering a numerical evaluation of any algorithms simply unfeasible. In order to compare the algorithms in scenarios with a high number of dimensions it will be required to tackle more efficiently the curse of dimensionality inherent to these scenarios using an approximate method to solve the associated CTMC. Nevertheless, solving the Markov chain with a lower precision will have an impact on the behaviour of the algorithms, which may vary from one to the other. A detailed study of the new behaviour of the algorithms in the presence of inaccurate solutions of the CTMC is outside the scope of this paper.

Furthermore, for the computation of the blocking probabilities, an additional fixed point iteration procedure is also required in order to obtain the value of the handover request rates (see the end of Section 7.3). At each iteration a multidimensional birth-and-death process must be solved. Solving this process, that in general will have a large number of states, constitutes the most computationally expensive part of the algorithm.

We make use of the same enhancement explained in [4] to eliminate the fixed point iteration to compute the handover arrival rates. Each run of `capacity` finds a $\lambda_{\max}^T$ (within tolerance limit) so that $\boldsymbol{p} \leq \boldsymbol{p}_{\mathbf{max}}$. Thus, instead of using (7.2) to compute λ_i^h we use the expression

$$\lambda_i^h = \lambda_i^n \frac{1 - B_i^n}{\mu_i^c/\mu_i^r + B_i^h} \tag{7.4}$$

Although (7.2) and (7.4) look very similar there is a substantial difference between the two. In Eq. (7.4), λ_i^h is explicitly defined whereas in (7.2) it is not as P_i^n and P_i^h depend on λ_i^h. Note that $\boldsymbol{p} = \boldsymbol{p}_{\mathbf{max}}$ only when $\lambda_{\max}^T$ is the system capacity (within tolerance limit), but using (7.4) reduces considerably the computation cost and therefore speeds-up the convergence rate of the algorithm.

We use expression (7.4) because by properly setting the configuration parameters of the MFGC policy it is possible to meet the QoS objectives with high precision (provided that there is a feasible solution). It is clear that when the aggregated arrival rate equals the system capacity then the value of the configuration parameters are such that the blocking probabilities perceived by the different streams are very close to their objectives. Therefore, even if the handover request rates computed in the beginning are imprecise, their precision improve as the algorithm progresses towards the maximum.

7.6 Numerical Evaluation

In this section we evaluate the computational complexity of our algorithm and compare it to the complexity of the HCO and PMC algorithms.

For the numerical examples we considered a system with two services ($N = 2$), and to assess the impact of mobility on computational complexity, five different scenarios (A, B, C, D, and E) were considered with varying mobility factors (μ_i^r/μ_i^c). The set of parameters that define scenario A are: $\boldsymbol{b} = (1, 2)$, $\boldsymbol{f} = (0.8, 0.2)$, $\boldsymbol{\mu_c} = (1/180, 1/300)$, $\boldsymbol{\mu_r} = (1/900, 1/1000)$, $\boldsymbol{B}^n = (0.02, 0.02)$, $\boldsymbol{B}^{ft} = (0.002, 0.002)$; all tolerances have been set to $\epsilon = 10^{-2}$. By (7.3), $\boldsymbol{B}^h \approx (0.01002, 0.00668)$ and then $\boldsymbol{p}_{\mathbf{max}} \approx (0.02, 0.02, 0.01002, 0.00668)$.

For the rest of scenarios the parameters have the same values as the ones used in scenario A except μ_i^r, which is varied to obtain four different mobility factor combinations: (B) $\mu_1^r = 0.2\mu_1^c$, $\mu_2^r = 0.2\mu_2^c$; (C) $\mu_1^r = 0.2\mu_1^c$, $\mu_2^r = 1\mu_2^c$; (D) $\mu_1^r = 1\mu_1^c$, $\mu_2^r = 0.2\mu_2^c$; (E) $\mu_1^r = 1\mu_1^c$, $\mu_2^r = 1\mu_2^c$.

A comparison of the number of floating point operations (flops) required by the HCO and PMC algorithms and our algorithm is shown in Table 7.2. The three algorithms were tested with the speed-up technique (see Section 7.5.1). It is worth noting that, as expected, the values obtained for the optimal capacity computed using the different methods were within tolerance in all tested cases.

Note that the HCO algorithm is provided with the prioritization order as input and therefore it does not need to search for it as their authors propose, which is a substantial advantage in terms of computation cost. Additionally, in its original version it does not implement the speed-up technique introduced in [4], without which the flops count is much higher than the one shown

Table 7.2 Comparison of the HCO (with known prioritization order) and PMC algorithms with our algorithm for different mobility factors (in Mflops).

	HCO				PMC				Our algorithm			
C	5	10	20	**TOT**	5	10	20	**TOT**	5	10	20	**TOT**
A	2.00	20.00	156.00	178.00	0.39	4.53	46.60	51.52	0.21	1.17	9.28	10.66
B	2.08	17.54	74.33	93.95	0.35	4.42	53.64	58.41	0.27	1.92	8.70	10.89
C	2.67	14.06	147.13	163.86	0.34	3.87	43.01	47.22	0.26	1.28	12.90	14.44
D	1.12	24.54	110.41	136.07	0.38	3.93	47.95	52.26	0.23	1.74	12.36	14.33
E	2.24	16.86	121.39	140.49	0.31	3.93	45.92	50.16	0.26	2.39	13.56	16.21
TOT	28.11	93.00	609.26	730.37	1.77	20.68	237.12	259.57	1.23	8.50	56.80	66.53

in Table 7.2. For example, for scenario A, the HCO algorithm with speed-up technique requires 2, 20 and 156 Mflops for $C = 5$, 10 and 20 respectively, while without the speed-up technique it needs 5.7, 60.2 and 438 Mflops, i.e. the speed-up technique divides the flop count by a factor of about three.

Our algorithm performs better than the other algorithms. The gain factor ranges from 4.9 to 17 with respect to the HCO algorithm and from 1.2 to 6.3 with respect to the PMC algorithm, with an average gain of 10.3 and 2.81 for HCO and PMC algorithms, respectively.

7.7 Conclusions

We proposed a new algorithm for computing the optimal setting of the configuration parameters for the Multiple Fractional Guard Channel (MFGC) admission policy in multiservice mobile wireless networks. The optimal configuration maximizes the offered traffic that the system can handle while meeting certain QoS requirements.

Compared to two recently published algorithms (HCO and PMC) ours, which is based on a simple and intuitive hill climbing approach, is less computationally expensive in all scenarios studied. Besides, the solution space concept discloses a novel insight into the problem of determining the optimal configuration parameter values of the MFGC policy, providing an heuristic evidence that the algorithm finds the optimal solution and converges in all scenarios.

Acknowledgements

This work has been supported by the Spanish Ministry of Education and Science (30%) and by the European Union (FEDER 70%) under projects TSI2005-07520-C03-03 and TSI2007-66869-C02-02. It is also supported by the “Catedra Telefonica” of the Universidad Politecnica de Valencia and Vodafone Spain.

D. Garcia-Roger was supported by the Generalitat Valenciana under contract CTB/PRB/2002/267.

References

[1] H. Heredia-Ureta, F. A. Cruz-Pérez and L. Ortigoza-Guerrero, Multiple fractional channel reservation for optimum system capcity in multi-service cellular networks, *Electronics Letters*, vol. 39, no. 1, pp. 133–134, January 2003.

[2] H. Heredia-Ureta, F. A. Cruz-Pérez and L. Ortigoza-Guerrero, Capacity optimization in multiservice mobile wireless networks with multiple fractional channel reservation, *IEEE Transactions on Vehicular Technology*, vol. 52, no. 6, pp. 1519–1539, November 2003.

[3] D. García, J. Martínez and V. Pla, Admission control policies in multiservice cellular networks: Optimum configuration and sensitivity, in *Mobile and Wireless Systems*, Lecture Notes in Computer Science (LCNS), vol. 3427, Springer-Verlag, Berlin/Heidelberg, pp. 121–135, 2005.

[4] V. Pla and V. Casares-Giner, Analysis of priority channel assignment schemes in mobile cellular communication systems: A spectral theory approach, *Performance Evaluation*, vol. 59, nos. 2-3, pp. 199–224, February 2005.

[5] S. Biswas and B. Sengupta, Call admissibility for multirate traffic in wireless ATM networks, in *Proceedings of IEEE INFOCOM*, vol. 2, pp. 649–657, 1997.

[6] F. Khan and D. Zeghlache, Effect of cell residence time distribution on the performance of cellular mobile networks, in *Proceedings of VTC’97*, IEEE, New York, pp. 949–953, 1997.

[7] V. Pla and V. Casares-Giner, Effect of the handoff area sojourn time distribution on the performance of cellular networks, in *Proceedings of IEEE MWCN*, September, pp. 401–405, 2002.

[8] P. V. Orlik and S. S. Rappaport, On the handoff arrival process in cellular communications, *Wireless Networks Journal (WINET)*, vol. 7, no. 2, pp. 147–157, March/April 2001.

[9] D. Hong and S. S. Rappaport, Traffic model and performance analysis for cellular mobile radio telephone systems with prioritized and nonprioritized handoff procedures, *IEEE Transactions on Vehicular Technology*, vol. VT-35, no. 3, pp. 77–92, August 1986. (See also: CEAS Technical Report No. 773, College of Engineering and Applied Sciences, State University of New York, Stony Brook, June 1999.)

[10] Y.-B. Lin, S. Mohan and A. Noerpel, Queueing priority channel assignment strategies for PCS hand-off and initial access, *IEEE Transactions on Vehicular Technology*, vol. 43, no. 3, pp. 704–712, August 1994.

[11] V. B. Iversen, The exact evaluation of multi-service loss systems with access control, in *Proceedings of the Teleteknik and Seventh Nordic Teletraffic Seminar (NTS-7)*, vol. 31, Lund (Sweden), pp. 56–61, August 1987.

[12] C.-T. Lea and A. Alyatama, Bandwidth quantization and states reduction in the broadband ISDN, *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 352–360, June 1995.

[13] B. Li, C. Lin and S. T. Chanson, Analysis of a hybrid cutoff priority scheme for multiple classes of traffic in multimedia wireless networks, *Wireless Networks Journal (WINET)*, vol. 4, no. 4, pp. 279–290, 1998.