Efficient Structured Policies for Admission Control in Heterogeneous Wireless Networks

Amin Farbod and Ben Liang

Department of Electrical and Computer Engineering
University of Toronto
Toronto, Ontario, CANADA

Email: {afarbod,liang}@comm.utoronto.ca

***Abstract*—In the near future, demand for Heterogeneous Wireless Networking (HWN) is expected to to increase. QoS provisioning in these networks is a challenging issue considering the diversity in wireless networking technologies and the existence of mobile users with different communication requirements. In HWNs with their increased complexity, "the curse of dimensionality" problem makes it impractical to directly apply the decision theoretic optimal control methods that are previously used in *homogeneous* wireless networks to achieve desired QoS levels. In this paper, optimal call admission control policies for HWNs are considered. A decision theoretic framework for the problem is derived by a dynamic programming formulation. We prove that for a two-tier wireless network architecture, the optimal policy has a two-dimensional threshold structure. Further, this structural result is used to design two computationally efficient algorithms, Structured Value Iteration and Structured Update Value Iteration. These algorithms can be used to determine the optimal policy in terms of thresholds. Although the first one is closer in its operation to the conventional Value Iteration algorithm, the second one has a significantly lower complexity. Extensive numerical observations suggest that, for all practical parameter sets, the algorithms always converge to the overall optimal policy. Further, the numerical results show that the proposed algorithms are efficient in terms of time-complexity and in achieving the optimal performance.**

***Index Terms*—Stochastic optimal control, quality of service, markov processes.**

I. Introduction

Heterogeneous Wireless Networking (HWN) is a major next-generation networking architecture to support ubiquitous wireless communications [1]. Current wireless communication technologies can generally be classified into two groups: local and global. Local services provide high-bandwidth and low latency communication services over a small area, while global services provide lower data rates to a wider area [2]. No single wireless communication technology is capable of simultaneously providing high bandwidth to a large number of mobile users over a wide area. HWN is a wireless networking paradigm to overcome this limitation. Such networks consist of several layers of different overlapping wireless networking technologies such as WiMAX/WiFi integration.

Corresponding Author: Ben Liang, Department of Electrical and Computer Engineering,University of Toronto,10 King's College Road, Toronto, Ontario, M4S 3G4, Email: liang@comm.utoronto.ca, Tel:+1-416-946-8614, Fax: +1-416-978-4425

This work was supported in part by a grant from LG Electronics and by Bell Canada through its Bell University Laboratories R&D program.

QoS provisioning in HWNs is a challenging issue considering the diversity of wireless networking technologies. Conventionally, call admission control (CAC) schemes are used in wireless networks to achieve a desired QoS level. A CAC algorithm decides to accept or reject call or handoff requests or to reserve resources in a resource-sharing systems. CAC schemes for *homogeneous* cellular networks have been extensively studied. These schemes can be classified into near-optimal heuristics [3] [4] and decision-theoretic optimal methods [5]–[7]. Furthermore, Dynamic Programming (DP) and Markov Decision Processes (MDP) [8] are used in the design of optimal CAC algorithms.

However, for almost all realistic modelings of networking systems, the computational load of finding an optimal policy by MDP algorithms is very high. Also, the size of state space grows exponentially with system capacity. Numerical methods [8] to solve MDP problems are iterative and as reported in [9], there is no known strongly-polynomial time algorithm to solve them. This can hinder the application of optimal CAC schemes in practical scenarios. As a remedy, one common modeling approach is to isolate one cell from the rest of the network to avoid excessive complexity in state space [10].

A more effective use of DP-based methods is to obtain structural results for optimal control problems [11]–[15]. In structural results, a DP formulation is used to characterize the structure of possible optimal policies. Then, knowledge of the policy structure can be exploited to design very efficient numerical methods to find the optimal policy. As an example, in [5], it is shown that the optimal control policy for a single cellular Base-Station (BS) is the well-known guard-channel policy. Then, knowing that the guard-channel policy is fully determined by a single threshold, the authors of [5] propose an efficient method to find it. It has been shown in the literature that for a large class of optimal control problems the optimal policy is threshold-based [5] [15].

To the best of our knowledge, there is no existing study on optimal CAC schemes for heterogeneous networks. Due to the increased complexity in HWN, direct application of MDP algorithms is impractical. In this paper, optimal CAC policies for HWNs are considered and some structural results are presented. We base our algorithms on theories in optimal control where dynamic programming methods are used to find the optimal policy to control a random process over time

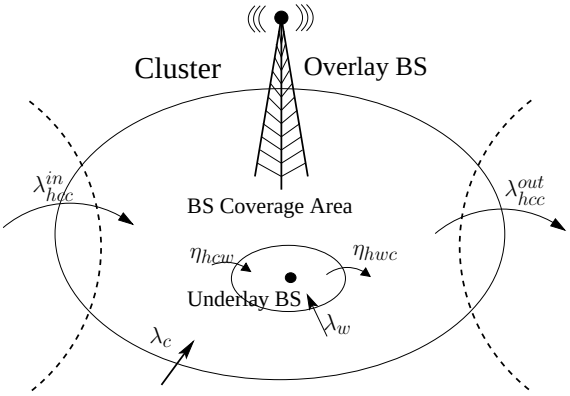

Fig. 1. Cluster traffic arrival and departure.

to achieve a certain optimization goal. A decision theoretic framework for the problem is derived by dynamic programming formulation. In this paper, we limit our focus to a two-tier wireless network architecture. With some modifications, this model can be applied to more complex scenarios. We prove that for this architecture the optimal policy has a two-dimensional threshold structure. Further, this structural result is used to design two computationally efficient algorithms, Structured Value Iteration (SVI) and Structured Update Value Iteration (SUVI). These algorithms can be used to determine the optimal policy in terms of thresholds. Although the first one is closer in its operation to the conventional Value Iteration (VI) algorithm, the second one has a significantly lower complexity. Extensive numerical observations suggest that, for all practical parameter sets, the algorithms always converge to the overall optimal policy.

The rest of the paper is organized as follows. In Section II, the system model and assumptions are presented. Section III presents the structural results and discussion on the complexity of algorithms to solve optimal CAC problems. In Section IV, the proposed algorithms are explained, and numerical results are given in section V, followed by concluding remarks in Section VI.

II. Network Model

A HWN can possibly have a complex configuration, involving many different wireless service layers. It is generally difficult to analytically tract such complicated scenarios to provide insight into the optimal control of resources in HWNs. In this work, we assume a two-tier heterogeneous wireless network architecture consisting of an overlay and an underlay. This basic 2-tier entity will be called a *Cluster*. An example cluster is shown in Fig. 1. There are also neighboring clusters from which horizontal handovers are possible to this cluster. We assume tight coupling between different layers of wireless network [16] in a cluster. In the tight coupling architecture, the management of different layers is centralized. In what follows, we assume that there exists a control unit which makes the CAC decision for the underlay and overlay BSs, and that clusters act independently and can measure the rate of external arrival processes such as hand-overs from neighbor clusters. Note that our mathematical analysis and control algorithms are independent of underlying wireless technologies as long as they satisfy some general technical requirements. However, in the simulation section parameters are chosen with respect to IEEE 802.16 WiMAX and IEEE 802.11 WLAN standards.

Service requests (more specifically calls in this work) arrive according to a memoryless Poisson process, and also service times are memoryless. Average service times are μ_c and μ_w for calls inside overlay and underlay. We also assume a memoryless mobility pattern where calls move to neighbor clusters or different layers at exponentially distributed times with rates given in Table I. It is clear that these assumptions result in exponential channel holding times [17]. This is an essential requirement in the application of MDP methods. In this paper, we focus on call-level QoS, which is common in CAC literature. Further, the fixed channel allocation (FCA) scheme is used and C_c and C_w denote the capacity of overlay and underlay in terms of the maximum number of calls they can accommodate. FCA easily applies to various wireless technologies with channel being frequency, time-slot or code assignment. Based on the results in [18] and [19], we consider the case where the number of available voice/multimedia channels in underlay/overlay can be quantized.

In our event-based DP, we associate costs to undesirable control decision events. These costs correspond to the dropping or blocking of arriving calls, and they are incurred when a call admission request is rejected by the cluster control unit. They reflect the degradation in the QoS from the service provider's perspective or the inconvenience of service denial perceived by users. The call and handoff arrival rates and their corresponding rejection costs are shown in Table I. Throughout the rest of this paper, every call type is called a *class*.

In the study of CAC schemes several optimality criteria are considered. The most common ones are minimization of a total cost (objective) function and minimization of the blocking probability given some hard constraints on dropping probabilities. In [5], these are refereed to as MINOBJ and MINBLOCK, respectively. The main advantage of MINBLOCK lies in the fact that it can guarantee some upper bounds on the dropping probabilities. This can also be achieved by MINOBJ by adjusting cost ratios. Furthermore, MINBLOCK has the drawback of not taking into account how much resource is wasted in reservation to achieve those bounds [10]. In this paper, we focus on MINOBJ for its flexibility. We can formally define MINOBJ as

$$\text{MINOBJ}: \quad \min\, g_\pi = \sum_{k=1}^{L} C_R^{(k)} \lambda_k P_B^{(k)} \qquad (1)$$

where $C_R^{(k)}$ is the cost of rejecting a call request of class k, λ_k is the arrival rate of class k calls, $P_B^{(k)}$ is the blocking (dropping) probability for that class and L is the total number of call classes.

III. Optimal CAC Policy

Decision theoretic optimization for Markovian processes is a well-known stochastic control method [20]. The Markov property allows significant reduction in tabular programming complexity and in some cases makes it possible to obtain structural results. An MDP is determined by four components:

#	Rate	Rejection Cost	Description
1	λ_c	C_{NBC}	New calls to Overlay
2	λ_w	C_{NBW}	New calls to Underlay
3	η_{hcc}	C_{HDCC}	Handoff to Overlay from Overlay
4	η_{hcw}	C_{HDCW}	Handoff to Overlay from Underlay
5	η_{hwc}	C_{HDWC}	Handoff to Underlay from Overlay

TABLE I
CALL ARRIVAL/HANDOFF RATES AND REJECTION COSTS.

state space S, action space A, state transition probabilities $P(.)$, and a cost function $C(.)$. The performance criteria can be formulated with respect to finite or infinite horizons and for average-cost or discounted-cost problems. The solution to an MDP is called a policy or rule. A policy maps the state space to actions $\pi : S \rightarrow A$, such that the optimization goal is achieved. A large class of policies, in which the decision is independent of time, are called stationary policies.

In this work, we wish to minimize the average expected cost for an infinite-horizon problem. This reflects our concern about long-run QoS performance. We start with a finite-horizon optimal cost function and we show that the solution to the infinite-horizon problem has the same structure. Let us denote by $V_k(i,j)$ the optimal cost function for a k-stage problem with the initial state (i,j) where i is the number of calls in overlay and j is the number of calls in underlay at the start of the decision epoch. Using the uniformization technique [21], we can write $V_{k+1}(i,j)$ recursively as

$$\begin{aligned}
V_{k+1}(i,j) = &\frac{\lambda_c}{v_{max}} \min(V_k(i,j) + C_{NBC}, V_k(i+1,j)) \\
&+ \frac{\lambda_w}{v_{max}} \min(V_k(i,j) + C_{NBW}, V_k(i,j+1)) \\
&+ \frac{\lambda_{hcc}^{in}}{v_{max}} \min(V_k(i,j) + C_{HDCC}, V_k(i+1,j)) \\
&+ \frac{i\eta_{hcw}}{v_{max}} \min(V_k(i-1,j) + C_{HDCW}, V_k(i-1,j+1)) \\
&+ \frac{j\eta_{hwc}}{v_{max}} \min(V_k(i,j-1) + C_{HDWC}, V_k(i+1,j-1)) \\
&+ \frac{i\mu_c}{v_{max}} V_k(i-1,j) + \frac{j\mu_w}{v_{max}} V_k(i,j-1) \\
&+ \frac{\lambda_{hcc}^{out}}{v_{max}} V_k(i-1,j) + (1 - \frac{v_{out}(i,j)}{v_{max}}) V_k(i,j) \qquad (2)
\end{aligned}$$

where $v_{out}(i,j)$ is the rate of going out of state $s = (i,j)$,

$$\begin{aligned}
v_{out}(i,j) = &\lambda_c + \lambda_w + \lambda_{hcc}^{in} + \lambda_{hcc}^{out} + i\eta_{hcw} \\
&+ j\eta_{hwc} + i\mu_c + j\mu_w, \qquad (3)
\end{aligned}$$

$\lambda_{hcc}^{out} = i\eta_{hcc}$, and v_{max} is the uniformization parameter such that $v_{max} \geq v_{out}(i,j)$ for every (i,j) pair. Since $v_{out}(i,j)$ is increasing in i and j, we choose $v_{max} = v_{out}(C_c, C_w)$. Equation (2) consists of nine terms, each reflecting one possible event; the first three terms reflect arrivals to the cluster, the fourth and fifth terms account for vertical handovers, the next three terms are for departure events and the last term is due to the uniformization technique where staying in the same state is possible. We also assume the following boundary conditions

$$\begin{aligned}
&V_k(C_c + 1, j) = \infty \text{ and } V_k(-1, j) = 0 \; 0 \leq j \leq C_w \\
&V_k(i, C_w + 1) = \infty \text{ and } V_k(-1, j) = 0 \; 0 \leq i \leq C_c. \qquad (4)
\end{aligned}$$

A. Optimality of Threshold-Based Policy

We show that the optimal policy to minimize the average cost for the system model given in Section II is a 2D threshold-based policy. In a single threshold system, that threshold is independent of the system state. When the system state is more complex, such as in the HWNs case, the threshold for the operation of one system component might depend on the state of another one. In our scenario, it gives rise to a 2D threshold structure.

From (2), it can be seen that when a call of class L arrives, it is only admitted if $V_k(i',j') - V_k(i,j) \leq C_R^L$, where state $s = (i,j)$ is the current state, state $t = (i',j')$ is the next state if we admit the call, and C_R^L is the rejection cost for class L. Let us define two difference operators for $V_k(i,j)$,

$$\begin{aligned}
\Delta^i V_k(i,j) &= V_k(i,j) - V_k(i-1,j) \\
\Delta^j V_k(i,j) &= V_k(i,j) - V_k(i,j-1). \qquad (5)
\end{aligned}$$

For every fixed j there is a sequence of $\Delta^i V_k(i,j)$ for $i = 1 \ldots C_c$, and vice versa. In what follows we claim that the sequences of $\Delta^i V_k(i,j)$ and $\Delta^j V_k(i,j)$ are increasing in i and j, respectively. For the proof refer to the Appendix.

Lemma 1: $V_k(i,j)$ is convex and monotonically non-decreasing in i (or j) for every fixed j (or i).

It has been shown in [21] that for average-cost problems with finite S and A, the optimal policy is stationary. Further, we are only interested in stationary policies which result in irreducible chains. The chain defined by $V_k(i,j)$ is also aperiodic since it contains loops into the same state. According to Theorem (6.6.2) in [21], for irreducible and aperiodic markov decision processes the difference of upper and lower bounds of $V_{k+1}(i,j) - V_k(i,j)$ converges to the optimal average cost per unit time when $k \rightarrow \infty$. Also, Theorem (6.6.1) in [21] implies that the optimal per-unit-time average cost function has the same structure as $V_k(i,j)$ defined in (2). Hence, structural results on $V_k(i,j)$ would directly hold for the infinite-horizon per-unit-time cost function.

Theorem 1: A 2D threshold-based policy is an optimal solution to the control problem with the system model given in (2).

Proof: Without loss of generality, let us assume that a call of class L arrives to overlay when the system state is $s = (i-1, j_0)$. The proof for arrivals to underlay is similar. If the call is admitted, increase in the optimal cost function is $\Delta^i V_k(i, j_0)$. We show that the CAC decision can be expressed in terms of thresholds determined by $\Delta^i V_k(i, j_0)$ and C_R^L.

From Lemma 1, we know that the sequence of $\Delta^i V_k(i, j_0)$ is increasing in i. If there is an $\hat{i}$ for which $\Delta^i V_k(\hat{i}, j_0) \leq C_R^L$ and $\Delta^i V_k(\hat{i}+1, j_0) > C_R^L$, then $\hat{i}$ is the threshold for admission to overlay when there are j_0 calls in underlay. Otherwise, if for every $\hat{i} \in \{1, \ldots, C_c\}$ we have that $\Delta^i V_k(\hat{i}, j_0) \leq C_R^L$ then that call is of high priority and it is only rejected when the system is full. Also, if for every $\hat{i}$, $\Delta V_{j_0}(\hat{i}, j) > C_R^L$ then

that call class is of low priority and it is never admitted to the system.

Note that for every call class of L, threshold $\hat{i}$ depends on $\Delta^i V_k(\hat{i}, j_0)$ which in turn depends on j_0. This implies that the threshold for overlay operation depends on the underlay state. Therefore, the optimal control policy has to be 2D threshold-based to account for this correlation. ■

B. CAC Algorithm

We denote by $\pi = \langle \text{Th}_c[C_w, M], \text{Th}_w[C_c, N] \rangle$ the class of threshold-based polices. Here, M is the number of call classes entering overlay and N is the number of call classes entering underlay. Every class within M or N would be called a subclass. In our scenario M is 3 and N is 2. The CAC algorithm when system state is $s = (i, j)$ at the arrival epoch and policy π is employed is given in Algorithm 1. When a call of subclass L' arrives to overlay (underlay), it is only admitted if the number of active calls in overlay (underlay) is less than the threshold for that call-type. This threshold is a function of call subclass and number of calls in the other network underlay (overlay).

A CAC algorithm is fully determined given policy π in terms of thresholds. However, finding these values is a non-trivial problem. Efficient computation of these thresholds is considered in the next section.

Algorithm 1 2D Threshold-Based CAC

Input: $\pi = \langle \text{Th}_c[C_w, M], \text{Th}_w[C_c, N] \rangle$
A Call of class L arrives
It belongs to subclass L'
Output: Admission Decision

```
if Arrival to overlay then
    if i < Th_c(j, L') then
        return Admit
    else
        return Reject
    end if
else {Arrival to underlay}
    if j < Th_w(i, L') then
        return Admit
    else
        return Reject
    end if
end if
```

C. Finding Policy π

A major requirement for CAC algorithms is their adaptivity to network traffic dynamics. Since this is generally achieved by periodically updating the admission policy, the algorithm computational load has to be minimal. Depending on the system size, the computation cost of solving a general MDP can be very high. Several methods such as Value Iteration (VI), Policy Iteration (PI) and Linear Programming (LP) methods are developed to solve general MDP problems [8].

According to [9], no strongly-polynomial algorithm is known for solving MDPs. Although MDPs can be solved by conversion to LP problems, polynomial-time algorithms for LP are inefficient and impractical. On the other hand, practical LP algorithms can result in exponential time-complexity in the worst case when used to solve MDPs. Consequently, there are no efficient and practical polynomial-time algorithms to solve MDPs. Therefore, the computation cost of finding thresholds for the optimal policy can be a burden if we use any of these techniques. However, when we know about the optimal solution structure, we might be able to exploit this knowledge to solve the problem more efficiently.

Generally, either direct or indirect methods can be employed to find the CAC parameters, i.e., policy thresholds. Direct methods require calculating the average cost for a given policy π_1. This can be done by modeling the system as a continuous markov chain (CTMC). Note that every MDP problem given a policy π_1 can be analyzed as a Markov chain. Then Gaussian elimination-like methods can be used to find state probabilities and to calculate the average cost. Once we have the average cost we can use methods such as multidimensional bisection search [22] to find the parameters that minimize it. The problem with this method is that for a two-tier network each having capacity n, the size of Markov chain state space would be $O(n^2)$ and Gaussian elimination would take $O((n^2)^3) = O(n^6)$.

However, as explained previously, CAC algorithms have to be light weight to be of any practical use. In indirect methods we avoid a direct evaluation of cost function. Instead we use an iterative approximation. Along with that, we use our prior knowledge of optimal policy structure to further improve the algorithm time-complexity.

IV. Efficient Computational Algorithms

In this section, we introduce efficient computational algorithms to find the optimal CAC policy. We first describe the conventional Value Iteration (VI) algorithm. We then propose two efficient algorithms called Structured Value Iteration (SVI) and Structured Update Value Iteration (SUVI). The basic principle of these algorithms is similar to VI. However, we use our prior knowledge of the optimal solution structure to eliminate unnecessary computations.

A. Conventional VI Algorithm

Conventional Value Iteration (VI) algorithm is based on the Bellman-Ford iterative equation [8],

$$V_n(s) = \min_{a \in A(s)} \{c_s(a) + \sum_{t \in S} P_{st}(a) V_{n-1}(t)\}. \qquad (6)$$

Note that this equation is backward in time, such that $V_0(.)$ is the cost at the end of the process. In every iteration $V_n(s)$ is calculated for $\forall s \in S$. Here, S is the state space, and $A(s)$ is the set of possible actions at state s. $P_{st}(a)$ is the transition probability of going form s to t having taken action a, and $c_s(a)$ is the cost of taking action a in state s. In our model, the system state has two components, the number of calls in overlay i and the number of calls in underlay j; $s = (i, j)$. For every incoming call, either new or hand-off, at any state two

actions are possible: accept (denoted by 1) or reject (denoted by 0); $A(s) = \{0, 1\}$.

To find the state transition probabilities $P_{st}(a)$, we use fictitious decision epochs [21]. The computational load of evaluating the cost function in every step highly depends on the density of the $P_{st}(a)$ matrix. When times between decision epochs are exponentially distributed we can reduce the computation cost by introducing fictitious decision epochs at which no real decision has to be made. These correspond to departure events when no action is taken. By this technique, at every decision epoch either real or fictitious, the system state can only change to adjacent states, making many terms in $P_{st}(a)$ zero. However, to keep track of the epoch type we have to extend the state space by one dimension. The increased computation cost due to this enlarged state-space is compensated by the reduction in the $P_{st}(a)$ density.

We define the new state variable to be a triple $s = (i, j, k)$. Here, k is the departure or arrival type. We already have 5 call types from Table I. We add a fictitious call event type of 0 which corresponds to call departures with a fictitious decision of $a = 0$ to be taken at departure events. In addition, since the decision epochs can be at any randomly distributed time, a Semi-Markov Decision Process (SMDP) model need to be used [21]. Again, we take the uniformization rate to be $v_{max} = v_{out}(C_c, C_w)$. Also, we have to determine $v_s(a)$, the rate of going out of state s having taken action a. Here we give the transition probabilities for some of the state-action combinations in terms of transition rates with $P_{st}(a) = \frac{q_{st}(a)}{v_s(a)}$ and $s = (i, j, k)$:

$$q_{st}(a=1) = \begin{cases} (i+1)(\eta_{hcc} + \mu_c) & t = (i, j, 0) \\ j\mu_w & t = (i+1, j-1, 0) \\ \lambda_c & t = (i+1, j, 1) \\ \lambda_w & t = (i+1, j, 2) \\ \lambda_{hcc}^{in} & t = (i+1, j, 3) \\ (i+1)\eta_{hcw} & t = (i+1, j, 4) \\ j\eta_{hwc} & t = (i+1, j, 5) \end{cases} \tag{7}$$

for $k \in \{1, 3\}$ and $v_s(a = 1) = v_{out}(i + 1, j)$ and $v_{out}(i, j)$ given in (3). Another example for $k = 4$ is

$$q_{st}(a=0) = \begin{cases} (i-1)(\eta_{hcc} + \mu_c) & t = (i-2, j, 0) \\ j\mu_w & t = (i-1, j-1, 0) \\ \lambda_c & t = (i-1, j, 1) \\ \lambda_w & t = (i-1, j, 2) \\ \lambda_{hcc}^{in} & t = (i-1, j, 3) \\ (i-1)\eta_{hcw} & t = (i-1, j, 4) \\ j\eta_{hwc} & t = (i-1, j, 5) \end{cases} \tag{8}$$

with $v_s(a) = v_{out}(i-1, j)$. Note that in the above, a hand-off request from overlay to underlay was initially rejected ($a = 0$) leaving only $i - 1$ calls in overlay at the start of the decision epoch. We specify the boundary conditions by defining

$$\begin{aligned} V_n(C_c + 1, j, k) &= \infty \quad \text{for all j and k} \\ V_n(i, C_w + 1, k) &= \infty \quad \text{for all i and k} \\ V_n(-1, j, k) &= 0 \quad \text{for all j and k} \\ V_n(i, -1, k) &= 0 \quad \text{for all i and k} \end{aligned} \tag{9}$$

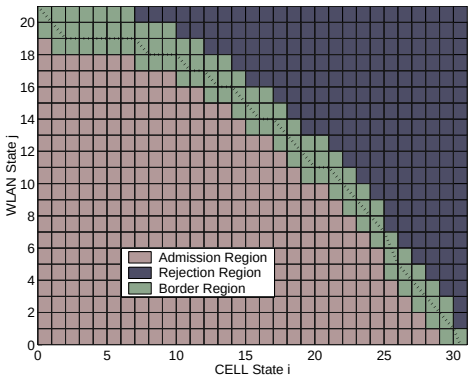

Fig. 2. SVI algorithm operation; AR, BR and RR for calls coming to underlay and $D = 1$.

For SMDP, (6) needs to be modified to reflect the semi-Markov state transition rates. We define operator $T_V[V(.), s, a]$ as

$$\begin{aligned} T_V[V(.), s, a] = & c_s(a) v_s(a) \\ & + \frac{v_s(a)}{v_{max}} \sum_{t \in S} P_{st}(a) V(t) \\ & + \left(1 - \frac{v_s(a)}{v_{max}}\right) V(s). \end{aligned} \tag{10}$$

Given this operator we can rewrite (6) for SMDPs as

$$V_n(s) = \min_{a \in A(s)} \{T_V[V_{n-1}, s, a]\}. \tag{11}$$

B. SVI Algorithm

Theorem 1 states that the optimal solution is a 2D threshold policy, implying that the admission region for any call type should be a closed area. An example of this is shown in Fig. 2. For any given policy π_1 and call subclass, we can partition the state space into three disjoint areas, called Accept-Region (AR), Border-Region (BR), and Reject-Region (RR). We define the region indicator function $I_R(s, p)$ for state $s = (i, j)$ and call request of subclass p as

$$I_R(s, p) = \begin{cases} \texttt{AR} & i - \text{Th}_c(j, p) < -D \\ \texttt{BR} & |i - \text{Th}_c(j, p)| \leq D \\ \texttt{RR} & i - \text{Th}_c(j, p) > D. \end{cases} \tag{12}$$

If a state is within distance D of the threshold level then it is in BR. D acts as a tuning parameter, determining the size of area we are willing to re-evaluate in every iteration. An example of $I_R(s, p)$ classification is shown in Fig. 2 for $D = 1$, where dotted states correspond to the threshold levels. The indicator function for the underlay subclasses is similar. Given the indicator function $I_R(s, p)$, we can redefine the action space $A(s)$ as $A'(s)$,

$$A'(s) = \begin{cases} \{0\} & \text{if } I_R(s, p) = \texttt{RR} \\ \{1\} & \text{if } I_R(s, p) = \texttt{AR} \\ \{0, 1\} & \text{if } I_R(s, p) = \texttt{BR}. \end{cases} \tag{13}$$

Here, we are limiting the set of possible actions. The idea is that for states inside the admission region it would be unnecessary to consider a possible reject action if they are not close to the border. Note that the cost function evaluation

is done iteratively until an optimum policy is reached. Thus, it can be expected that if an optimum action is not currently taken in a state, the algorithm will eventually reach that state and would choose the appropriate action. As we will see in the next section, extensive numerical observations suggest that, for all practical parameter sets under consideration, the algorithm always converges to the overall optimal policy.

Finally, the proposed SVI algorithm is given in Algorithm 2 as an extension to VI [21]. As in the VI algorithm, ϵ determines the desired accuracy. Note that in Algorithm 2 the new action space $A'(s)$ is used to improve the algorithm efficiency.

Algorithm 2 Structured Value Iteration Algorithm

Input: π, P_{st}, $c_s(a)$, D, v_{max}, ϵ
Output: Optimal Admission Policy

1: **Initialize** $V_0(s)$
   s.t. $\forall s \in S : 0 \leq V_0(s) \leq \min_a\{c_s(a)v_s(a)\}$
   $n := 0$
   $\forall s \quad \pi_0(s) = 1$
2: $n := n + 1$
3: **Find** $\forall s \in S$
   $V_n(s) = \min_{a \in A'(s)}\{T_V[V_{n-1}, s, a]\}$
   $\pi_n(s) = \arg\min_{a \in A'(s)}\{T_V[V_{n-1}, s, a]\}$
4: **Compute**
   $m_n = \min_{t \in S}\{V_n(t) - V_{n-1}(t)\}$
   $M_n = \max_{t \in S}\{V_n(t) - V_{n-1}(t)\}$
5: $\forall j, l \quad \text{Th}_c^n[j, l] = \arg\min_i\{\pi_n(i, j, l) = 0\}$
   $\forall i, l \quad \text{Th}_w^n[i, l] = \arg\min_j\{\pi_n(i, j, l) = 0\}$
   Recompute $A'(s)$ based on Th_c^n and Th_w^n
6: **if** $M_n - m_n \leq \epsilon m_n$ **then**
7: Policy π_n is optimal and Th^n is the threshold set
8: **Stop**
9: **else**
10: **Go to** step 2
11: **end if**

C. SUVI Algorithm

The intuition behind SVI was that it is not necessary at every point to evaluate the optimal cost function for all possible actions when some actions are very unlikely to be the optimal decision. In SVI, we assign a default action to every such point based on the region it belongs to, and then in every round of iteration the cost function for that point is updated according to that default action. However, we believe that some of these default updates might be non-necessary if there has not been a major decision or cost change in their neighborhood. Also, under the operation of a numerical algorithm similar to SVI, in every iteration the changes in cost or decisions can only happen within the border region. Therefore, it might be more efficient if we limit the scope of the cost function update to a neighborhood of the border region. More formally, let us define the Update Region (UR) indicator function $I_U(s, p)$ as

$$I_U(s,p) = \begin{cases} 1 & |i - \text{Th}_c(j,p)| \leq D_u \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

A state belongs to the UR if it is within distance D_u ($D_u \geq D$) of the threshold levels. An example of $I_U(s, p)$ classification

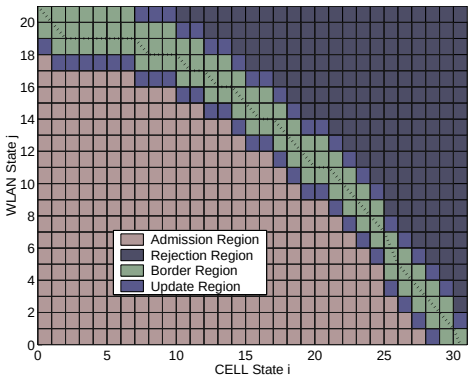

Fig. 3. SUVI algorithm operation; AR, BR, UR and RR for calls coming to underlay when $D = 1$ and $D_u = 2$.

Parameter	Value	Parameter	Value
C_c	50 calls	μ_c	0.01 sec^{-1}
C_w	20 calls	μ_w	0.01 sec^{-1}
λ_c	0.5 calls/sec	C_{NBC}	5
λ_w	0.1 calls/sec	C_{NBW}	3
η_{hcc}	5×10^{-3} sec^{-1}	C_{HDCC}	50
η_{hcw}	0.01 sec^{-1}	C_{HDCW}	10
η_{hwc}	0.02 sec^{-1}	C_{HDWC}	30

TABLE II
ARRIVAL RATES AND REJECTION COSTS USED IN SIMULATION.

is shown in Fig. 3 for $D = 1$ and $D_u = 2$. We define the action space $A''(s)$ as

$$A''(s) = \begin{cases} \{0\} & \text{if } I_R(s,p) = \texttt{RR} \text{ and } I_U(s,p) = 1 \\ \{1\} & \text{if } I_R(s,p) = \texttt{AR} \text{ and } I_U(s,p) = 1 \\ \{0,1\} & \text{if } I_R(s,p) = \texttt{BR} \\ \varnothing & \text{otherwise.} \end{cases} \tag{15}$$

The algorithm proposed for SVI in the last section can be used in conjunction with this new action space $A''(s)$ to find the optimal policy.

V. NUMERICAL RESULTS

The performance of the proposed methods is studied in this section. We use a combination of a C++ discrete event simulator and a MATLAB numerical program for that purpose. First, an optimal policy is found through iterative numerical calculations in MATLAB. Then, it is fed into the discrete-event simulator to find the resultant QoS performance. All parameters take their default values shown in Table II unless otherwise stated. For the rest of this section, we define the rejection cost vector $\overline{C}_R = \{C_R^{(1)}, .., C_R^{(5)}\}$, where $C_R^{(k)}$ is the rejection cost for call requests of class k as classified in Table I.

A. The Optimal Policy

The quality of the control policies obtained by using the proposed algorithms and their abilities to achieve the optimal performance is considered in this section. We conducted an extensive number of experiments to compare the result of

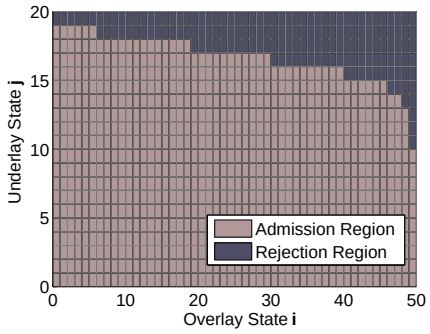

Fig. 4. ARRs for class 2.

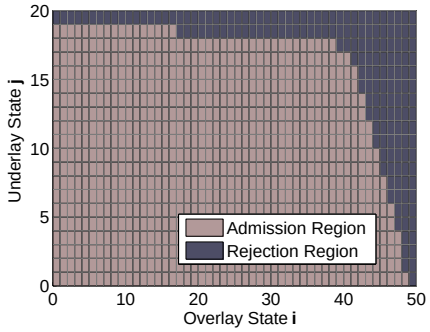

Fig. 5. ARRs for class 2; increased mobility.

SVI and SUVI with conventional VI. All comparisons of the threshold levels obtained by using SVI/SUVI and VI showed complete matching. These observations suggest that, for all practical parameter sets, the algorithms always converge to the overall optimal policy. Hence, they can be considered as reliable methods to perform CAC for HWNs.

Figure 4 illustrates the ARRs for the optimal policy obtained from SVI. It is assumed that $C_c = 50$ and $C_w = 20$. The ARRs for call class of 2 (new arrival to underlay) are depicted. Figure 5 shows the same results for another configuration in which the mobility rate η_{hwc} has increased from 0.02 to 0.08. Comparison of these figures shows that at the lower mobility rate, the rejection of calls is more evenly distributed over the set of states. Also, it can be observed that in both figures nearly the same percentage of states are in the rejection area. This is due to the fact that the prioritization of calls in SVI is mostly based on cost ratios defined in Table I, rather than the mobility pattern.

B. Convergence Speed

In section III, we explained that the complexity of CAC algorithms can impose limitations on their practicality. We compare the convergence speed of SVI and SUVI against VI for different values of D and D_u. Figures 6 and 7 show the Time Gain, $T^X_{Gain} = T_{VI}/T_X$, for SVI and SUVI over VI for a range of network capacities. The run time of each algorithm is obtained by using MATLAB to record its CPU time. In this experiment, C_c varies from 20 to 100 while maintaining $C_w = \frac{1}{3}C_c$. For every C_c, λ_c is chosen such that

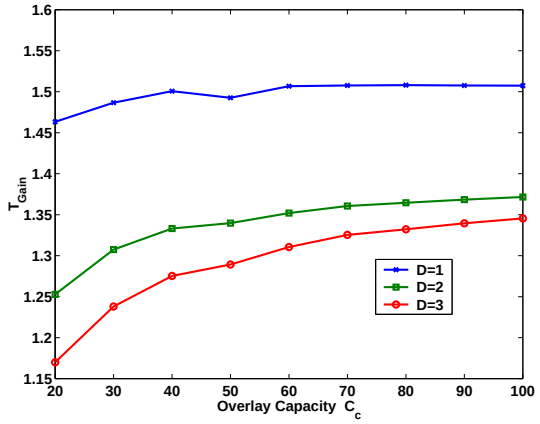

Fig. 6. SVI convergence speed gain T_{Gain} for $D = 1, 2, 3$.

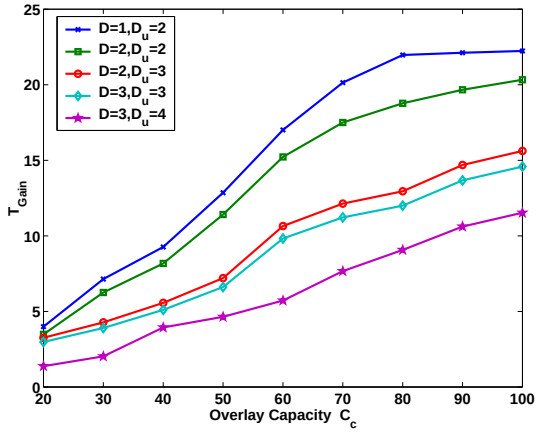

Fig. 7. SUVI convergence speed gain T_{Gain} for $D = 1, 2, 3$ and $D_u = 2, 3, 4$.

the nominal overlay utilization , $\rho_c = \frac{\lambda_c}{C_c \mu_{c_j}}$, is equal to 1, and $\lambda_w = 0.2 \times \lambda_c$. Also, we assume $\epsilon = 10^{-3}$.

As the network capacity increases, the ratio of BR area, in which full optimization is performed, relative to the area of AR/RR regions, in which a default action is evaluated, decreases. This accounts for the increased efficiency with the increase in network capacity. Figure 7 shows T_{Gain} for the SUVI algorithm. Each curve gives the time gain of the SUVI over VI for different network capacities for a given D and D_u. It can be observed that smart selection of update points can significantly improve the convergence speed. It is interesting to note that the best-performing parameters are $D = 1$ and $D_u = 2$, while the algorithm always converges to the optimal policy regardless of the selected tuning parameters.

C. Optimal Policy vs. Complete Sharing (CS)

A Complete Sharing (CS) policy refers to the admission policy in which $\pi(s) = 1$, regardless of the system state as long as it remains within system capacity boundaries; in our scenario $s = (i, j) \leq (C_c, C_w)$. In what follows, we compare the performance of the optimal policy obtained from SUVI (with $D = 1$ and $D_u = 2$) with the CS policy. CS policy essentially represents the performance of a system in which no proactive resource allocation scheme is employed

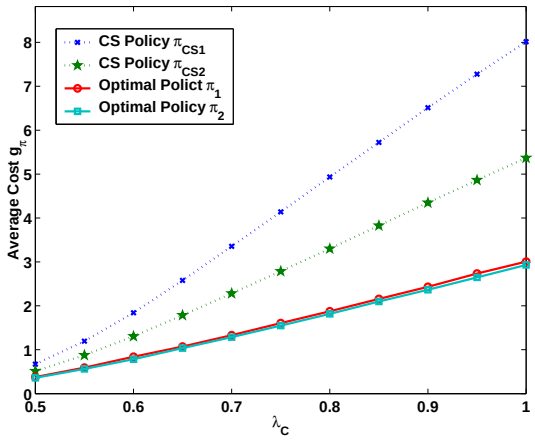

Fig. 8. Optimal and CS policy costs versus incoming traffic.

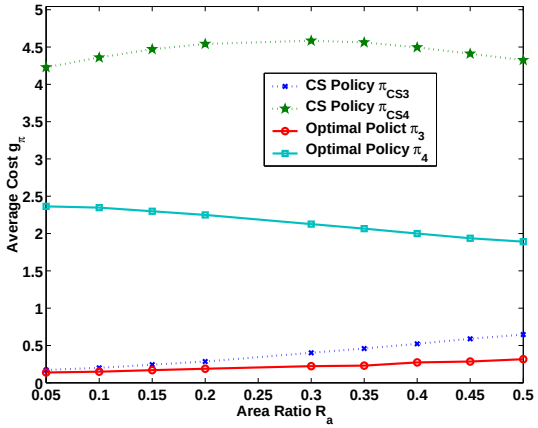

Fig. 9. Optimal and CS policy costs versus underlay area.

and call request are satisfied as long as there are enough resources. Comparing the optimal policy with the CS policy shows how effective the control algorithm is and what degrees of performance gain can be expected by adopting a non-trivial policy.

Figure 8 shows the result of this comparison for two different cost vectors, $\overline{C}_R$, when λ_c varies from 0.5 to 1.0. λ_c is chosen such that the nominal overlay utilization, ρ_c, ranges from 1.0 to 2.0. Also, we have $C_c = 50$, $C_w = 20$ and $\lambda_w = 0.2 \times \lambda_c$. In this experiment, we consider two cases having different cost vectors of $\overline{C}_{R1} = \{5, 3, 50, 10, 30\}$ and $\overline{C}_{R2} = \{5, 3, 20, 10, 20\}$. First, the optimal policies, π_1 and π_2, are computed for each cost vector for the given system parameters, and then a discrete event simulation is performed to obtain the resultant average cost g_π. The system is then exposed to the same load/cost settings under a CS policy to find the corresponding g_{cs}. Note that $g_\pi = \sum_{k=1}^{5} C_R^{(k)} \lambda_k P_B^{(k)}$.

Several insights can be gained from this figure. Let us define the cost improvement difference as $Q(\lambda) = g_{cs}(\lambda) - g_\pi(\lambda)$ and the cost reduction as $Y(\lambda) = \frac{g_{cs}(\lambda)}{g_\pi(\lambda)}$. First, $Q(\lambda)$ invariably increases when system is exposed to higher loads, which implies the control algorithm is more useful at higher loads. An interesting observation is that the cost reduction, $Y(\lambda)$, is higher for $\overline{C}_{R1}$ compared to the cost reduction for $\overline{C}_{R2}$.

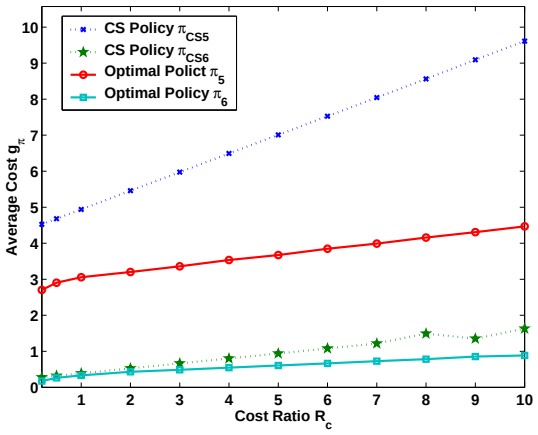

Fig. 10. Optimal and CS policy costs versus rejection cost ratio.

This is due to the fact that $\overline{C}_{R1}$ assigns a more diverse set of rejection costs to call classes. The closer the rejection costs are, the lower the cost reduction is going to be, and in the extreme case of having equal rejection costs, i.e., $C_R^{(k)} = C_0$ for $k = 1, ..., 5$, both CS and optimal policy yield the same average cost and $Y(\lambda)$ will have its minimum value of 1.

Figure 9 shows the comparison results for the case when the area ratio R_a of the underlay to overlay increases from 0.05 to 0.5. Let us assume that the total arrival rate to the cluster, consisting of overlay and underlay, is $\lambda_T = \lambda_c + \lambda_w$, and depending on the area's expanse covered by each layer, a fraction R_a of λ_T is assigned to the underlay while $1 - R_a$ goes to the overlay. Under user assignment strategies in which a user is initially assigned to the underlay if it is within the double-coverage area (overlap of underlay and overlay), R_a is directly proportional to the underlay expanse. Two cases are considered in this experiment. Policy π_3 is obtained for $\lambda_T = 0.5$ and π_4 is obtained for $\lambda_T = 1$, and also $\overline{C}_R = \{5, 3, 20, 20, 20\}$.

It can be observed from Fig. 9 that the cost difference $Q(R_a)$ is higher when the load is higher. One notable observation is that the average cost for the optimal policy π_4 decreases when R_a increases. This can be associated with the lower rejection cost for new arrivals to the underlay. When R_a increases, a larger number of incoming calls are assigned to underlay which if rejected accumulate a lower cost as compared to their rejection cost at the overlay. The non-monotonic trend in the average cost for CS policy π_{CS4} can be explained similarly.

Another set of average cost results is shown in Fig. 10. The cost ratio R_c used in this figure is defined as $R_c = \frac{C_{HDCW}}{C_{NBW}}$. Also, the rejection cost vector is $\overline{C}_{R1} = \{5, 5, 20, 5 \times R_c, 20\}$. Policy π_5 is computed for $\lambda_c = 0.5$ and policy π_6 is computed for $\lambda_c = 1$, while maintaining $\lambda_w = 0.2 \times \lambda_c$. Again, the higher the load, the larger is the difference between the average cost induced by the optimal and CS policies. A trend in this figure which requires explanation is the difference in rates at which average costs for the optimal and CS policies increases when R_c increases. In contrary to the expectation that the cost reduction $Y(R_c)$ should have remained fixed, as it was the case in Fig. 8, we see that $Y(R_c)$ is increasing in R_c. The reason is

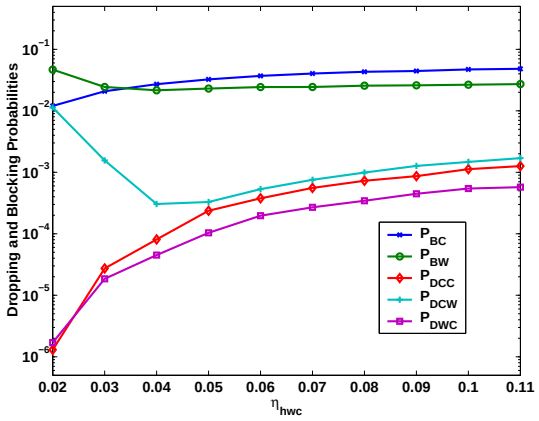

Fig. 11. Drop/block probabilities for increased mobility.

that when we increase R_c the blocking/dropping probabilities which constitute g_π remain constant under the CS policy, and hence, the average cost grows proportionally, whereas under the optimal policy these probabilities are adaptively changing to achieve the minimum possible average cost for any given R_c. The overall result is that the optimal control scheme does not allow for a linear change in system rejection costs to be reflected severely in the average cost.

D. QoS Performance

In this section, the control policy generated by SUVI is used to find the resultant performance. We use the policy obtained in the last section for $\eta_{hwc} = 0.02$. The system is exposed to an increased mobility rate η_{hwc} to study its response. Figure 11 shows the results. It can be observed that the dropping probabilities for calls of classes 3 and 4, P_{DCC} and P_{DWC}, are increasing. The increase of total incoming load to overlay has also elevated the blocking probability for new arrivals P_{BC}. More interestingly, the change in P_{DCW} is not monotonic, and it initially decreases and then increases. The initial decrease happens when the increase of η_{hwc} results in lowered levels of load in the underlay while overlay load has not significantly changed. The increasing part can be correlated to the situation in which many calls in overlay, including previous handovers from underlay, try to handover to the underlay.

VI. Conclusion

In this paper, optimal CAC for HWN is considered. A decision theoretic framework for the problem is derived by a dynamic programming formulation. Structural results on the optimal cost function for a two-tier HWN architecture are presented. It is shown that for such networks the cost function is convex. This is used to prove that the optimal CAC policy is two-dimensional threshold based. Then, efficient numerical methods called Structured Value Iteration (SVI) and Structured Update Value Iteration (SUVI) are proposed to determine the optimal admission policy. Although the first one is closer in its operation to the conventional Value Iteration algorithm, the second one has a significantly lower complexity.

Extensive simulation and numerical studies show that SVI and SUVI are both reliable and effective. Firstly, they always converge to the optimal policy in much less time compared to conventional numerical methods used to solve MDPs. Also, discrete-event call-level simulations confirm that the obtained policy is effective in maintaining the desired QoS performance.

The proposed method is a new framework for systems with complex state descriptions. It can potentially be used in stochastic control of queuing systems. In general, under some minor technical assumptions, the same method can be applied to many problems involving optimal control for stochastic systems with multidimensional state-space.

References

[1] M. Stemm and R. H. Katz, "Vertical Handoffs in Wireless Overlay Networks." *ACM/Kluwer Mobile Networks and Applications (MONET)*, vol. 3, no. 4, 1998.

[2] A. H. Zahran, B. Liang, and A. Saleh, "Signal threshold adaptation for vertical handoff in heterogeneous wireless networks," *ACM/Kluwer Mobile Networks and Applications (MONET), Special Issue on Soft Radio Enabled Heterogeneous Networks*, vol. 11, no. 4, pp. 625–640, 2006.

[3] M. Naghshineh and M. Schwartz, "Distributed call admission control in mobile/wireless networks," *IEEE JSAC*, vol. 14, no. 4, pp. 711–717, 1996.

[4] S. Wu, K. Wong, and B. Li, "A dynamic call admission policy with precision QoS guarantee using stochastic control for mobile wireless networks," *IEEE/ACM Trans. on Networking*, vol. 10, no. 2, pp. 257–271, 2002.

[5] R. Ramjee, D. Towsley, and R. Nagarajan, "On optimal call admission control in cellular networks," *Wireless Networks*, vol. 3, no. 1, pp. 29–41, 1997.

[6] C.-J. Ho and C.-T. Lea, "Improving call admission policies in wireless networks," *Wireless Networks*, vol. 5, no. 4, pp. 257–265, 1999.

[7] J. Choi, T. Kwon, Y. Choi, and M. Naghshineh, "Call admission control for multimedia services in mobile cellular networks: A markov decision approach," in *Fifth IEEE Symposium on Computers and Communications,Proc.ISCC 2000.*, 2000, pp. 594–599.

[8] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc. New York, NY, USA, 1994.

[9] M. Littman, T. Dean, and L. Kaelbling, "On the complexity of solving Markov decision problems," in *Proc. of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence*, 1995, pp. 394–402.

[10] Z. Haas, J. Halpern, L. Li, and S. Wicker, "A decision-theoretic approach to resource allocation in wireless multimedia networks," in *Proc. of the 4th ACM International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, 2000, pp. 86–95.

[11] E. Altman, T. Jimenez, and G. Koole, "On optimal call admission control in resource-sharing system," *IEEE Trans. on Communications*, vol. 49, no. 9, pp. 1659–1668, 2001.

[12] J. Ni, D. H. K. Tsang, S. Tatikonda, and B. Bensaou, "Optimal and structured call admission control policies for resource-sharing systems," *IEEE Trans. on Communications*, pp. 158–170, 2007.

[13] A. Gavious and Z. Rosberg, "A restricted complete sharing policy for a stochastic knapsack problem in b-isdn," *IEEE Trans. on Comm.*, vol. 42, no. 7, pp. 2375–2379, 1994.

[14] K. Ross and D. Tsang, "The stochastic knapsack problem," *IEEE Trans. on Communications*, vol. 37, no. 7, pp. 740–747, 1989.

[15] R. Nobel and H. Tijms, "Optimal control of a queueing system with heterogeneous servers and setup costs," *IEEE Trans. on Automatic Control*, vol. 45, no. 4, 2000.

[16] V. Varma, S. Ramesh, K. Wong, M. Barton, G. Hayward, and J. Friedhoffer, "Mobility management in integrated umts/wlan networks," in *IEEE Int. Conf. on Comm.(ICC)*, 2003, pp. 1048–1053.

[17] Y. Fang, I. Chlamtac, and Y.-B. Lin, "Channel occupancy times and handoff rate for mobile computing and pcs networks," *Computers, IEEE Trans. on*, vol. 47, no. 6, pp. 679–692, 1998.

[18] K. Medepalli, P. Gopalakrishnan, D. Famolari, and T. Kodama, "Voice capacity of IEEE 802.11b, 802.11a and 802.11g wireless LANs," in *IEEE Global Telecomm. Conf.(GLOBECOM)*, 2004, pp. 1549–1553.

[19] D. Hole and F. Tobagi, "Capacity of an ieee 802.11b wireless lan supporting voip," in *IEEE Int. Conf. on Comm.*, 2004, pp. 196–201.

[20] S. M. Ross, *Applied Probability Models with Optimization Applications*. Dover Publications, 1992.
[21] H. C. Tijms, *A First Course in Stochastic Models*. John Wiley and Sons Ltd., 2003.
[22] G. R. Wood, "The bisection method in higher dimensions," *Journal of Mathematical Programming*, vol. 55, pp. 319–337, 1992.

APPENDIX

Proof of Lemma 1: $V_{k+1}(i,j)$ as shown in (2) consists of 9 terms, $T_1 \dots T_9$. Here for clarity, we factor out the common $1/v_{max}$ term and label the costs by their class number. We define

$$D_m(i,j) = T_m(i,j) - T_m(i-1,j) \tag{16}$$

$$V_{k+1}(i,j) = \frac{1}{v_{max}} \sum_{m=1}^{9} T_m(i,j) \tag{17}$$

$$\Delta^i V_{k+1}(i,j) = \frac{1}{v_{max}} \sum_{m=1}^{9} D_m(i,j). \tag{18}$$

We use induction on k to prove $V_{k+1}(i,j)$ is convex and monotonically non-decreasing in i for every fixed j (proof for convexity in j is similar). This is equal to showing $\Delta^i V_{k+1}(i+1,j) \geq \Delta^i V_{k+1}(i,j)$ and $\Delta^i V_{k+1}(i,j) \geq 0$. The induction hypothesis is the following: *$V_k(i,j)$ is (I) monotonically non-decreasing in i and j and (II) convex in i(or j) for every fixed j(or i).*

Proof of (I): We show $\Delta^i V_{k+1}(i,j) \geq 0$ by evaluting each D_m term individually. The basis step is trivial since $\forall (i,j) \leq (C_c, C_w) : V_0(i,j) = 0$ and $\forall j : V_0(C+1,j) = \infty$. Consider $D_1(i,j)$, when λ_c is factored out. Let

$$\begin{aligned}
&D_1'(i,j) = [(T_1(i,j) - T_1(i-1,j)]/\lambda_c \\
&= \min(V_k(i,j) + C_1, V_k(i+1,j)) \\
&\quad - \min(V_k(i-1,j) + C_1, V_k(i,j)) \\
&= \underbrace{\min(C_1, V_k(i+1,j) - V_k(i,j))}_{\geq 0 \quad \text{by hypo.(I) for } V_k} + V_k(i,j) \\
&- \min(C_1, V_k(i,j) - V_k(i-1,j)) - V_k(i-1,j) \\
&\geq V_k(i,j) - V_k(i-1,j) \\
&\quad - \min(C_1, V_k(i,j) - V_k(i-1,j)) \\
&\geq 0 \quad \text{(by hypo.(I) for } V_k\text{)}.
\end{aligned} \tag{19}$$

The same method can be applied directly to $T_{\{2,3,5,7\}}$. However, the rest of the terms have to be considered altogether. We first extend $D_9(i,j)$ as follows,

$$\begin{aligned}
&D_9(i,j) = T_9(i,j) - T_9(i-1,j) \\
&= (v_{max} - v_{out}(i,j))V_k(i,j) \\
&- (v_{max} - v_{out}(i-1,j))V_k(i-1,j) \\
&= (C_c - i)\eta_{hcc}V_k(i,j) - (C_c - i + 1)\eta_{hcc}V_k(i-1,j) \\
&+ (C_c - i)\eta_{hcw}V_k(i,j) - (C_c - i + 1)\eta_{hcw}V_k(i-1,j) \\
&+ (C_w - j)\eta_{hwc}V_k(i,j) - (C_w - j)\eta_{hwc}V_k(i-1,j) \\
&+ (C_c - i)\mu_c V_k(i,j) - (C_c - i + 1)\mu_c V_k(i-1,j) \\
&+ (C_w - j)\mu_w V_k(i,j) - (C_w - j)\mu_w V_k(i-1,j).
\end{aligned} \tag{20}$$

Although it is straight forward to show that the 3rd and 5th terms above are non-negative, for other terms it is not trivial. We show that every of these terms in $D_9(i,j)$ if combined with other terms in $\Delta^i V_k(i,j)$ can be proved to be non-negative. As an example, consider the first term which we can rewrite as

$$\begin{aligned}
&D_9^{\eta_{hcc}} = (C_c - i)\eta_{hcc}V_k(i,j) - (C_c - i + 1)\eta_{hcc}V_k(i-1,j) \\
&= (C_c - i)\eta_{hcc} \underbrace{\{V_k(i,j) - V_k(i-1,j)\}}_{\Delta^i V_k(i,j) \geq 0} - \eta_{hcc}V_k(i-1,j).
\end{aligned} \tag{21}$$

Equation 21 contains η_{hcc}, and from (2) we see that term T_8 has η_{hcc} as well, so we will use $D_8(i,j)$. We consider the sum of them:

$$\begin{aligned}
&D_9^{\eta_{hcc}} + D_8(i,j) = \\
&(C_c - i)\eta_{hcc}\{V_k(i,j) - V_k(i-1,j)\} - \eta_{hcc}V_k(i-1,j) \\
&\quad + i\eta_{hcc}V_k(i-1,j) - (i-1)\eta_{hcc}V_k(i-2,j) \\
&= (C_c - i)\eta_{hcc}\{V_k(i,j) - V_k(i-1,j)\} \\
&\quad + (i-1)\eta_{hcc}\{V_k(i-1,j) - V_k(i-2,j)\} \\
&= (C_c - i)\eta_{hcc}\Delta^i V_k(i,j) + (i-1)\eta_{hcc}\Delta^i V_k(i-1,j) \\
&\geq 0.
\end{aligned} \tag{22}$$

The other two terms in $D_9(i,j)$ can be matched with T_4 and T_6 similarly. Hence, $\Delta^i V_{k+1}(i,j) = \sum_{m=1}^{9} D_m \geq 0$.

Proof of (II): We need to show $\Delta^i V_{k+1}(i+1,j) \geq \Delta^i V_{k+1}(i,j)$. This is equal to $\sum_{m=1}^{9}[D_m(i+1,j) - D_m(i,j)] \geq 0$. Similar to the last part, the basis step is trivial since $\forall (i,j) \leq (C_c, C_w) : V_0(i,j) = 0$ and $\forall j : V_0(C+1,j) = \infty$. Let us define $Y_m(i,j) = D_m(i,j) - D_m(i-1,j)$. Again, we prove terms are non-negative either individually or when combined with other terms. Let us start with Y_1, having factored out λ_c:

$$\begin{aligned}
&Y_1'(i+1,j) = [D_1(i+1,j) - D_1(i,j)]/\lambda_c \\
&= \min(C_1, V_k(i+2,j) - V_k(i+1,j)) + V_k(i+1,j) \\
&\quad - \min(C_1, V_k(i+1,j) - V_k(i,j)) - V_k(i,j) \\
&\quad - \min(C_1, V_k(i+1,j) - V_k(i,j)) - V_k(i,j) \\
&\quad + \min(C_1, V_k(i,j) - V_k(i-1,j)) + V_k(i-1,j) \\
&= \underbrace{\min(C_1, \Delta^i V_k(i+2,j)) - \min(C_1, \Delta^i V_k(i+1,j))}_{\geq 0 \quad \text{by hypo.(II) for } V_k} \\
&\quad + \Delta^i V_k(i+1,j) - \Delta^i V_k(i,j) \\
&\quad - \big[\min(C_1, \Delta^i V_k(i+1,j)) - \min(C_1, \Delta^i V_k(i,j))\big] \\
&\geq \Delta^i V_k(i+1,j) - \Delta^i V_k(i,j) \\
&\quad - \big[\min(C_1, \Delta^i V_k(i+1,j)) - \min(C_1, \Delta^i V_k(i,j))\big] \\
&\geq 0 \quad \text{(by hypo.(II) for } V_k\text{)}.
\end{aligned} \tag{23}$$

Again, the proof for $Y_{\{2,3,5,7\}}$ is the same as for Y_1. For other terms, we have to take an approach similar to the last part. Let us extend Y_9, focusing on the terms containing η_{hcc}:

$$\begin{aligned}
&Y_9(i+1,j) = D_9(i+1,j) - D_9(i,j) \\
&= \big(v_{max} - v_{out}(i+1,j)\big)V_k(i+1,j) \\
&\quad - \big(v_{max} - v_{out}(i,j)\big)V_k(i,j) \\
&- \big\{\big(v_{max} - v_{out}(i,j)\big)V_k(i,j)
\end{aligned}$$

$$
\begin{aligned}
&\quad - \big(v_{max} - v_{out}(i-1,j)\big)V_k(i-1,j)\big\} \\
&= (C_c - i - 1)\eta_{hcc}V_k(i+1,j) - (C_c - i)\eta_{hcc}V_k(i,j) \\
&- (C_c - i)\eta_{hcc}V_k(i,j) + (C_c - i + 1)\eta_{hcc}V_k(i-1,j) \\
&+ \dots . \qquad (24)
\end{aligned}
$$

Separating the terms with η_{hcc}, we obtain

$$
\begin{aligned}
&Y_9^{\eta_{hcc}}(i+1,j) = \\
&(C_c - i - 1)\eta_{hcc}V_k(i+1,j) - (C_c - i)\eta_{hcc}V_k(i,j) \\
&- (C_c - i)\eta_{hcc}V_k(i,j) + (C_c - i + 1)\eta_{hcc}V_k(i-1,j) \\
&= (C_c - i - 1)\eta_{hcc}\left\{V_k(i+1,j) - V_k(i,j)\right\} \\
&\quad - (C_c - i - 1)\eta_{hcc}\left\{V_k(i,j) - V_k(i-1,j)\right\} \\
&\quad - 2\,\eta_{hcc}V_k(i,j) + 2\,\eta_{hcc}V_k(i-1,j). \qquad (25)
\end{aligned}
$$

This term has to be evaluated along with $Y_8(i+1,j)$ in order to show that the sum of both terms is non-negative. For $Y_8(i+1,j)$ we have

$$
\begin{aligned}
&Y_8(i+1,j) \\
&= (i+1)\eta_{hcc}V_k(i,j) - i\eta_{hcc}V_k(i-1,j) \\
&\quad - i\eta_{hcc}V_k(i-1,j) + (i-1)\eta_{hcc}V_k(i-2,j) \\
&= (i-1)\eta_{hcc}\left\{V_k(i,j) - V_k(i-1,j)\right\} \\
&\quad - (i-1)\eta_{hcc}\left\{V_k(i-1,j) - V_k(i-2,j)\right\} \\
&\quad + 2\,\eta_{hcc}V_k(i,j) - 2\,\eta_{hcc}V_k(i-1,j), \qquad (26)
\end{aligned}
$$

and the sum of these two terms is

$$
\begin{aligned}
&Y_9^{\eta_{hcc}}(i+1,j) + Y_8(i+1,j) = \\
&= (C_c - i - 1)\eta_{hcc}\left\{V_k(i+1,j) - V_k(i,j)\right\} \\
&\quad - (C_c - i - 1)\eta_{hcc}\left\{V_k(i,j) - V_k(i-1,j)\right\} \\
&\quad + (i-1)\eta_{hcc}\left\{V_k(i,j) - V_k(i-1,j)\right\} \\
&\quad - (i-1)\eta_{hcc}\left\{V_k(i-1,j) - V_k(i-2,j)\right\} \\
&= (C_c - i - 1)\eta_{hcc}\underbrace{\left\{\Delta^i V_k(i+1,j) - \Delta^i V_{k+1}(i,j)\right\}}_{\geq 0 \quad \text{by hypo.}(II)\text{ for } V_k} \\
&\quad + (i-1)\eta_{hcc}\underbrace{\left\{\Delta^i V_k(i,j) - \Delta^i V_{k-1}(i,j)\right\}}_{\geq 0 \quad \text{by hypo.}(II)\text{ for } V_k} \\
&\geq 0. \qquad (27)
\end{aligned}
$$

It is similar to show non-negativity for the other terms of Y_m. Since we have $\forall m : Y_m \geq 0$, the inequality $\Delta^i V_{k+1}(i+1,j) \geq \Delta^i V_{k+1}(i,j)$ holds, and hence, the cost function is convex. ■