Distributed Load Balancing Algorithm for Adaptive Channel Allocation for Cognitive Radios

Simon Fischer*, Marina Petrova†,Petri Mähönen†, Berthold Vöcking*
* RWTH Aachen University, Computer Science 1, 52056 Aachen, Germany
† RWTH Aachen University, Department of Wireless Networks, Kackertstrasse 9, 52072 Aachen, Germany
Email: fischer@cs.rwth-aachen.de

***Abstract*— The problem of channel allocation has been extensively studied in the context of cellular networks. There is a substantial amount of work in the field of dynamic frequency assignment for WLANs and mesh networks as well. The growing interest in the cognitive radio technology and its capability to offer more efficient spectrum usage brought this problem back as one of the most popular research topics nowadays. In this paper we present two quite simple to implement, distributed algorithms for selecting channels, in cognitive radio environment, so that the load is distributed (smoothed) over them.**

I. Introduction

Cognitive radio (CR) technology is emerging as an opportunity to enable dynamic and real-time spectrum access. The cognitive radio paradigm was originally introduced by Mitola [1]. Cognitive radios can determine the best-possible radio settings, through a learning process using the environmental stimuli. This enables the radio to achieve optimal performance objectives. CRs offer an advanced spectrum management capability and are currently extensively studied as a solution to enhance spectrum utilization.

The problem of channel allocation in wireless networks has been an interesting research topic for long time now. A large number of solutions have been suggested for the cellular networks and WLANs. Techniques such as simulated annealing, graph colouring, neural networks and tabu search has been considered [2], [3], [4], [5]. An excellent review on various solutions for frequency assignment in cellular networks is given in [6]. Similarly, there are number of solutions, both centralized and distributed, addressing the frequency allocation problem in WLANs [7], [8], [9].

Recently, efforts have been made to model the opportunistic spectrum access and solve the channel allocation in a cognitive radio environment using a game theoretical approach, see [10], [11], [12] as examples. We believe that there are, in fact, two separate channel allocation problems related to cognitive radio networks. First and the obvious one is the much studied problem for the secondary users to detect spectrum opportunities that are opened by the non-transmitting primary users. In this case, the secondary users need to find out the spectrum opportunities and decide how to use them. Second, there is a problem how secondary users should chose between the available channels. This is, of course, a sort of load balancing or resource allocation problem, which can be quite difficult to solve if there is no central authority and the environment becomes very dynamic. In fact, Mähönen and Petrova [13] have noted that both the first and second problem maybe open to *flash crowd* effects in certain circumstances.

This paper addresses the problem of frequency selection by secondary users in a CR environment. The goal is to minimize the interference level and maximize the throughput for each individual user without damaging the performance of the other users. We present two simple algorithms to achieve load balancing among the channels for the secondary users in quite general framework. The algorithms are sampling based and as such reach equilibrium quite fast.

The paper is organized as follows. In section II we describe our system model. Section III and IV we give an overview of the load balancing algorithms that we are using for adaptive channel allocation. Section VI presents our simulation results and observations. finally in section VII we conclude our paper.

II. System Model

We consider the following system model. We are given n balls (*agents* with a cognitive radio capability) and m bins (channels or frequencies). For simplicity, we assume that all balls have equal size, i. e., all users impose the same traffic load to the system and they use the same radio technology for communication, e.g., WLAN (802.11b/g). Each radio is assigned with a frequency, and n_i denotes the *load* of channel i, i. e., the number of balls that select bin $i \in [m]$. Since we assume that agents are indistinguishable, the system can be fully described by the state vector $\mathbf{n} = (n_i)_{i\in[m]}$. The equal traffic load is selected to keep our discussion simple. The presented solution will also work with unequal loads.

Each bin i has an individual capacity bound δ_i which may or may not be elastic. We consider only systems with elastic capacity boundaries such as CDMA (Code Division Multiple Access). Systems with fixed capacity boundaries such as TDMA (Time Division Multiple Access) are not considered here.

The ball should decide if it is staying in the bin or changing the bin based on the objective to achieve the best possible performance. It should be noted that the balls have a Poissonian living-time. It can happen that new balls (users) come into the system and try to acquire a channel. This means that there will be fluctuations of the number of balls n in time.

Each bin $i \in [m]$ is associated with a *cost function*[1] $c_i : \mathbb{N} \mapsto \mathbb{R}_0^+$. The value of $c_i(n_i)$ specifies the cost incurred to all balls choosing bin i. For the cost function we may choose any of the utilities described below with the convention that users aim at minimizing these values, as is the case, e. g., for latency. If the utility is any quantity which is to be maximized, e.g., throughput, then we can model cost as its reciprocal. We normalize the scale at which we measure cost, such that the maximum value of any c_i is 1 and the minimum value is 0. Let

$$C(\mathbf{n}) = \sum_{i \in [m]} \frac{n_i}{n} \cdot c_i(n_i)$$

denote the average cost sustained by the agents. The natural question to ask in this case is: What are the balanced states in this model? A ball has an incentive to migrate from its current bin to another bin if, it gains more utility by doing so. We can consider a state $\mathbf{n}$ to be stable, if no agent has such an incentive. This notion is formalized by the concept of Nash equilibria.

Definition 1 (Nash equilibrium): A state $\mathbf{n}$ is at a *Nash equilibrium* if for all machines i and j with $n_i > 0$ it holds that $c_i(n_i) \leq c_j(n_j + 1)$.

Due to a potential function argument [14], pure Nash equilibria do always exist in this model, even if balls have different weights [15]. Our goal is the design of distributed algorithms by which balls can attain such an equilibrium in an exact or in an approximate sense quickly. One should note that Nash equilibria do not necessarily optimize the overall performance of the system, but they have the appealing property of being fair from a local point of view, thus ensuring stability.

Before defining the utility functions for the balls in a single bin we will first name several assumptions we take into account. We assume that there is no inter-channel interference in the system. This means that the effect of overlapping channels is neglected. Furthermore the balls have the capability to sample the number of balls n_i in a certain bin. Generally speaking it is very difficult to know how many radios are using the same frequency in the IEEE 802.11 case for example. We are of course able to measure radiation power and by making an educated guess estimate the n_i. This is only a weak assumption, as we will be showing in this paper that the precise knowledge is not necessary for our distributed channel allocation algorithm.

We can now define general utilities that can be used as a goal to play the game and come up to a algorithmic solution for adaptive frequency allocation.

1) *Utility 1*: Minimize interference. The goal is to minimize the number of balls per bin. In a more complicated case when an inter-channel interference is considered we should minimize the number of n_i in a single bin but also in the neigbouring $(i-1)$ and $(i+1)$ bins.
2) *Utility 2*: Maximize throughput. Maximizing the throughput is tightly related to minimizing interference. In case of a stochastic MAC less balls will lead both to a minimized interference and maximized throughput.
3) *Utility 3*: Minimize Latency. This may be a valid selection criterion for time-sensitive applications like streaming or voice calls.

[1]In this paper we mostly use terms utility and cost interchangeably. In the game theoretical concepts and algoritm description we are using the term cost function, however, the networking community is often using utility function.

In this paper we consider the throughput maximization as a utility function using two different MAC protocols, namely slotted-ALOHA and CSMA/CA as example cases.

The slotted ALOHA [16] is a well-known extension of the pure ALOHA where the users are allowed to transmit only at the beginning of the slot. The maximum throughput that can be achieved is $S = Ge^{-G}$, where the G is an offered load. It should be noted that in our case all the users offer the same amount of traffic load according to a same Poisson process.

In order to consider CSMA/CA and to have a functional form for the utility, we have used the well-known Bianchi's analytical model for the saturation throughput of IEEE 802.11 DCF systems. We refer the reader to look details from [17], where the complex derivation is well described. The functional form has been used to test our game model, but the detailed parameter values are not used. This is due to fact that the functional forms of cost functions are, in our case, a defining factor for the performance of the algorithm, and the specific values of different multipliers do not affect the convergence properties. We omit the detailed description of the Bianchi approximation equations due to space limitation.

III. Computation Model

In the following, we will present two algorithms that satisfy the properties which we believe are crucial for the implementation of real-world load adaptive channel allocation algorithms in a decentralized manner.

- *Local control.* Algorithms are executed locally by wireless devices. They do not rely on central coordination by a base station or any selected node which takes the role of a leader or coordinator. In particular, they do not rely on the assumption that other nodes execute the same algorithm.
- *Local information.* Algorithms may rely exclusively on information which they are able to gather or sense autonomously.
- *Simplicity.* Computation executed at the individual nodes is simple and stateless. This ensures that nodes can join and leave the network at any time without re-initialization.
- *Efficiency.* Energy consumptions should be reduced to minimum.
- *Selfishness.* In order to be sure that our protocols are accepted by the users, they must strive to maximize their own utility rather than the overall system utility.

In general, we assume that an agent using channel i can easily determine its own cost $c_i(n_i)$ (e. g., by measuring its throughput). As mentioned before, we assume that CRs can estimate, by scanning the spectrum, the number of agents using each channel. To that end, we provide our algorithms with a primitive MEASURE_LOAD(i) which returns the value of n_i. We will also take into account that this value may be observed under the influence of uncertainty.

To decide whether or not an agent should switch from its current channel to another, it would be helpful to know the utility of the candidate destination channel. Though this utility can be measured by migrating to this channel on a trial basis, such a test may be prohibitively expensive. Alternatively, the CR may decode packet headers on the target channel to estimate the utility perceived by agents currently using this channel. Independent of a particular implementation, we provide our algorithms with a second primitive MEASURE_COST(i) which returns $c_i(n_i)$. As a first approach we will present an algorithm that makes use of this primitive, subsequently showing how we can implement a similar effect by purely relying on the primitive MEASURE_LOAD(i).

Finally, each agent maintains a variable $channel$ which determines the channel currently used.

IV. Load Balancing Algorithms

A. The Utility-Aware Scenario

In this section, we assume that MEASURE_COST() can be applied to any channel. Consider an agent currently using channel i. At intervals, this agent performs two steps. First, it determines the vector $\mathbf{n}$ using MEASURE_LOAD() and then randomly samples another agent. Now, the agent compares its own cost with the cost of channel j used by the sampled agent. If $c_j < c_i$, it migrates towards this channel with probability $c_i - c_j$. Pseudocode of algorithm COMPARE_AND_BALANCE is given in Algorithm 1.

Algorithm 1 COMPARE_AND_BALANCE

```
for all balls in parallel do
    $c \leftarrow$ MEASURE_COST($channel$)
    for all channels $i \in [m]$ do
        $n_i \leftarrow$ MEASURE_LOAD($i$)
    end for
    $n \leftarrow \sum_{i\in[m]} n_i$
    choose $j$ randomly with $\mathbb{P}[j] = n_j/n$
    $c' \leftarrow$ MEASURE_COST($j$)
    if $c' < c$ then
        with probability $c - c'$: $channel \leftarrow j$
    end if
end for
```

At first sight, this algorithm might look counterintuitive. The probability of a channel being sampled increases with the number of agents already utilizing it. This seems to contradict the actual goal of balancing the load. However, there are fundamental theoretical reasons for this approach.

Intuitively, the reason for sampling other agents is that we take the observation that many agents utilize a channel as an indicator that this channel has a high utility. To make that more precise, assume, for the time being, that our algorithm knew an assignment $\mathbf{n}^*$ at a Nash equilibrium in advance. In that case, it would be optimal to sample channel i with probability n_i^*/n since this would yield a Nash equilibrium within one round in expectation. Since $\mathbf{n}^*$ is, in general, not known in advance, we must find an estimator for it. Suppose that cost functions are linear, i. e., $c_i(n_i) = n_i/s_i$ where we envision the parameter s_i as the *speed* of channel i. Since at a Nash equilibrium, the values of $c_i(n_i)$ are approximately equal, n_i^* is approximately proportional to s_i which suggests to sample channels proportionally to their speed. In fact, this technique has been successfully applied in [18] in a discrete model, and in [19], [20] in a continuous model. However, the value of s_i may not be known, or it may not even exist for nonlinear cost functions. In this case, we may use n_i as an estimator for n_i^*. This estimator becomes better and better quickly as the system approaches a balanced state.

There is a second reason, why load proportional sampling is useful. Assume that we sample channels with some static probability which does not depend on its current load or utility. Then, there exists a channel which has sampling probability at most $1/m$. For this reason, any bound on the time to reach or even approximate a Nash equilibrium must be at least linear in m, since any ball may need expected time at least m to find a better channel. This is proven in precise way in [20].

B. The Utility-Unaware Scenario

In the preceding section we have used the primitive MEASURE_COST(j) in order to estimate the utility of the sampled channel j. We now assume that this primitive can only be applied to the channel currently used by the agent. Instead of comparing the utilities of two channels, the behavior of the following algorithm depends only on the utility of the currently used channel. At intervals, an agent observes its own cost c_i and decides to move to another channel with a probability that increases with c_i. Again, the target channel is sampled proportionally to n_i. The algorithm, called AVOID_CONTENTION, is specified in pseudocode in Algorithm 2.

Algorithm 2 AVOID_CONTENTION

```
for all balls in parallel do
    $c \leftarrow$ MEASURE_COST($channel$)
    measure own cost $c$
    with probability $c$:
    for all channels $i \in [m]$ do
        $n_i \leftarrow$ MEASURE_LOAD($i$)
    end for
    $n \leftarrow \sum_{i\in[m]} n_i$
    choose $j \in [m]$ randomly where $\mathbb{P}[j] = n_j/n$
    $channel \leftarrow j$
end for
```

Compared to algorithm COMPARE_AND_BALANCE, algorithm AVOID_CONTENTION is more energy efficient. Not only does it refrain from invoking MEASURE_COST on channels other than the one currently used, also the measurement of

$\mathbf{n}$ by the primitive MEASURE_LOAD is executed only when the decision to switch to a new channel has already been made. In COMPARE_AND_BALANCE this measurement must be executed as the first step in each round.

The energy efficiency of the second algorithm comes at some cost. Whereas algorithm COMPARE_AND_BALANCE certainly stabilizes as soon as a Nash equilibrium is reached, algorithm AVOID_CONTENTION does not. At a Nash equilibrium, there will still be some fluctuations. However, the expected load vector that results from one round starting at a Nash equilibrium is at a Nash equilibrium again. To see this, consider a load vector $\mathbf{n}$. Since any ball in bin j migrates to bin i with probability $c_j(n_j) \cdot n_i/n$, the expected load n'_i of any channel i after one step is

$$\begin{aligned} \mathbb{E}[n'_i] &= n_i - n_i \cdot c_i(n_i) + \sum_{j\in[m]} n_j \cdot c_j(n_j) \cdot \frac{n_i}{n} \\ &= n_i - n_i \cdot c_i(n_i) + n_i \cdot C(\mathbf{n}) \qquad (1) \\ &= n_i \ . \end{aligned}$$

The latter equality holds since at a Nash equilibrium, $c_i(n_i) = C(\mathbf{n})$ (or $n_i = 0$). It is easy to check that this property is not preserved if the sampling probabilities used by algorithm AVOID_CONTENTION are modified.

Let us remark that for both algorithms, once we have $n_i = 0$ for some bin i, the sampling probability for bin i is zero, and hence the bin will stay empty. This problem can be easily avoided, e. g., by adding one virtual agent to every bin, i. e., to use a load vector n' with $n'_i = n_i + 1$. The event of a bin becoming empty, however, is so improbable that it never occurred during our simulations. We therefore omit this modification from our description and simulations for clarity.

V. ANALYSIS IN THE FLUID LIMIT

Let us consider the two algorithms described above for the case that the number of users $n \to \infty$, the so-called fluid limit. Rather than considering the number of balls n_i in bin i we now consider the fraction $x_i = n_i/n$ of balls in bin i. By the law of large numbers, as $n \to \infty$, we identify random variables with their expectation. The cost functions $c_i(\cdot)$ generalize naturally in this model by extending their domain to the interval $[0, 1]$. We consider the time derivative $\dot{x}_i$ at which the fraction of balls in bin i changes. For algorithm AVOID_CONTENTION we have seen in Equation (1) that

$$\dot{x}_i = x_i \cdot (C(\mathbf{x}) - c_i(x_i)) \ .$$

What about algorithm COMPARE_AND_BALANCE? Here, the probability to migrate from a channel i to a channel j with $c_j < c_i$ is $n_j/n \cdot (c_i(n_i) - c_j(n_j))$. In the fluid limit,

$$\begin{aligned} \dot{x}_i &= \sum_{j:c_j(x_j)>c_i(x_i)} x_j \cdot x_i \cdot (c_j(x_i) - c_i(x_i)) \\ &\quad - \sum_{j:c_j(x_j)<c_i(x_i)} x_i \cdot x_j \cdot (c_i(x_i) - c_j(x_j)) \\ &= \sum_{j\in[m]} x_j \cdot x_i \cdot (c_j(x_i) - c_i(x_i)) \\ &= x_i \cdot (C(\mathbf{x}) - c_i(x_i)) \ . \end{aligned}$$

Interestingly, we obtain the same expression for $\dot{x}_i$ for both algorithms in the fluid limit.

In [19] it is shown that this dynamics converges towards a Nash equilibrium and reaches a state in which at most an ϵ-fraction of the agents differs by more than an ϵ-fraction from the average cost in time

$$\mathcal{O}\left(\frac{1}{\epsilon^3} \cdot \log\left(\frac{\max_{\mathbf{x}} C(\mathbf{x})}{\min_{\mathbf{x}} C(\mathbf{x})}\right)\right) \ .$$

In [21], [20] this analysis is improved by taking into account the fact that information may be out of date by the time it is used. Thus, due to concurrent action of the agents, overshooting and oscillation effects may occur. Still, it can be shown that convergence can be guaranteed if the policy is executed slowly enough.

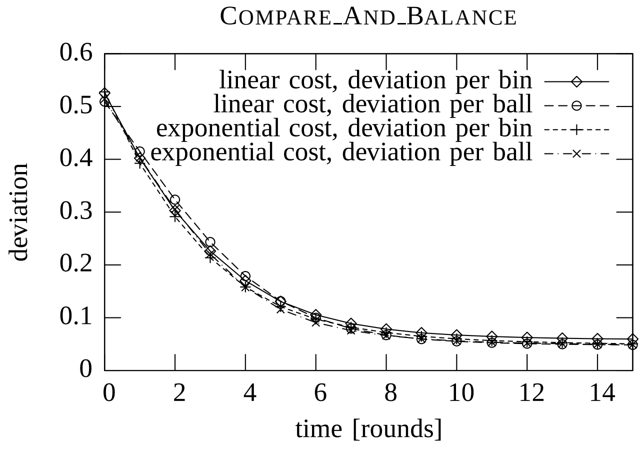
VI. SIMULATION

Analyses of these algorithms in the fluid limit are somewhat optimistic since the fact that the number of agents is finite introduces effects that are not present in the fluid limit. Berenbrink *et al.* [22] analyze a similar algorithm in a discrete round based model with identical linear cost functions. For this special case, their analysis yields an upper bound of $\mathcal{O}\left(\log\log n + m^4\right)$ rounds to reach a Nash equilibrium.

A. Convergence

Simulations show, that also for other cost functions, the algorithms presented above converge very quickly. We have performed simulations of both algorithms for $n = 500$ and $m = 10$ using both linear and exponential cost functions. We have performed a series of $10,000$ runs for each combination of algorithm and cost function type. Each run starts with a random assignment of agents to channels and consists of 15 iterations. The parameters a_i of the cost functions $c_i(n_i) = a_i \cdot n_i$ and $c_i(n_i) = a_i \cdot \exp(n_i \cdot (m/n))$ were chosen uniformly at random from the interval $[1, 10]$. For each iteration, we considered the random variables C_c and C_a where C_c is the cost of a channel chosen uniformly at random and C_a is the cost experienced by an agent chosen uniformly at random. We use the standard deviation of these random variables as a measure of the balancedness of the system.

Figure 1 shows how the standard deviation (after normalizing the expectation to be 1) decreases over time. We see that the minimum standard deviation is reached after as little as 6 rounds for both algorithms. As we expected, algorithm COMPARE_AND_BALANCE comes close to a Nash



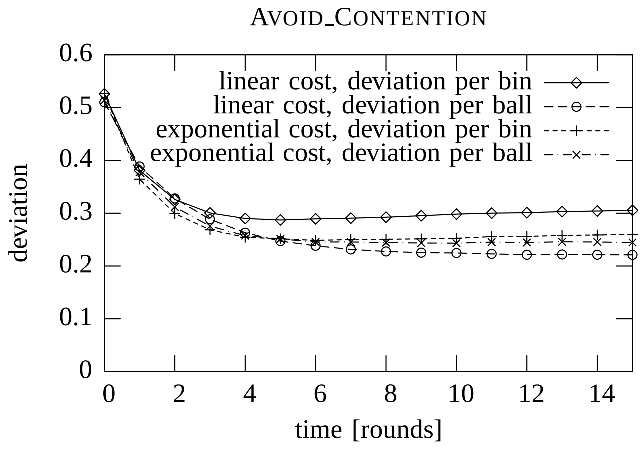


Fig. 1. Relative standard deviation of the cost values for the algorithms COMPARE_AND_BALANCE and AVOID_CONTENTION.

equilibrium. The deviation from the average cost decreases to below 6%. As we expected, algorithm AVOID_CONTENTION performs worse. The expected deviation from the average cost decreases only to around 25%. This, however, is still by a factor of more than 2 better than the deviation of the random initial assignment.

Our simulations also suggest that the behavior of the algorithms is largely independent of the class of cost functions.

B. Dependence on Number of Users

The approximation quality achieved by algorithm AVOID_CONTENTION depends on the number of users in the system. As the number of users increases, the number of users utilizing a particular channel becomes more and more concentrated around its expectation. This is illustrated in Figure 2.

Again, we have performed $1,000$ runs with $m = 10$ and n varying between 10 and 250, i. e., the number of users per channel varying between 1 and 25. We see that, as n increases, the relative deviation from the average cost decreases.

C. Observation Under Uncertainty

In the previous section we have assumed that the primitives MEASURE_LOAD(i) and MEASURE_COST(i) return the exact

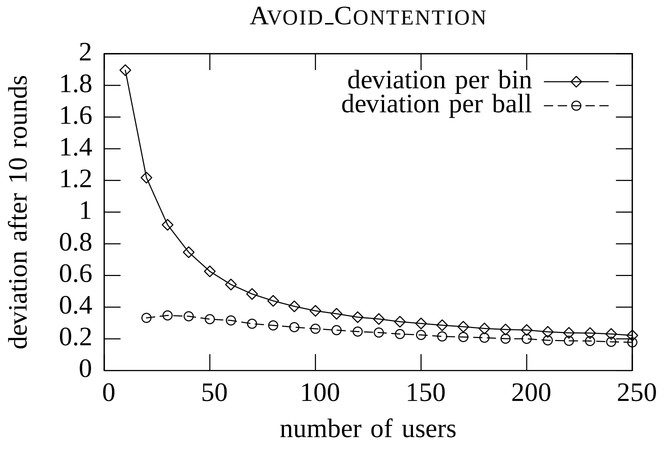


Fig. 2. Relative standard deviation after 10 rounds depending on the number of users.

load and cost of the given bin. We now assume that load and cost are observed under the influence of uncertainty. More precisely, we assume that the above primitives return a value that is drawn uniformly at random from the interval $[(1-\eta)\cdot x, (1+\eta)\cdot x]$ where x is the true value (n_i or $c_i(n_i)$, respectively) and η is the error parameter which we vary between 0 and 1. For each value of η we perform three times $1,000$ simulations adding uncertainty either to load, cost, or both parameters.

Figure 3 shows the results of our simulations. The plots show the relative deviation from the average cost after 10 rounds of the algorithm. We see that the average deviation from the average cost achieved by both algorithms increases linearly with the amount of uncertainty under which cost is observed. This is hardly surprising since we cannot expect an algorithm to achieve an amount of balancedness that is beyond what can be observed by the agents.

To determine the correlation between the amount of uncertainty and the balancedness achieved by the algorithms, let $d(\eta)$ denote the average deviation from the average cost after 10 rounds if the cost $c_i(n_i)$ is observed under the influence of uncertainty with parameter η as described above. Then, using a least-squares regression for $\eta \in [0.5, 1]$, we obtain

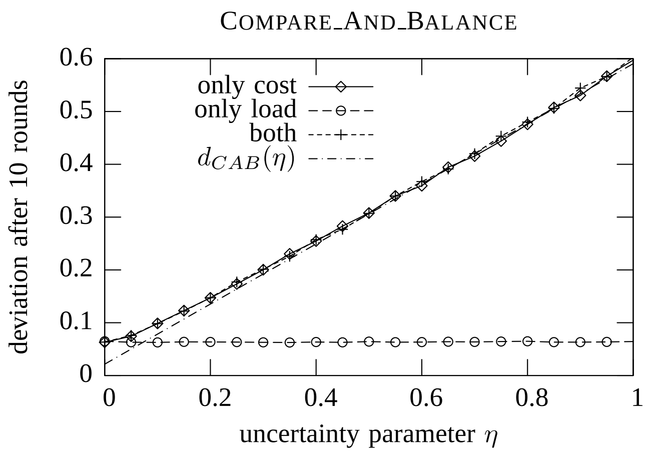
$$d_{AC}(\eta) \approx 0.520 \cdot \eta + 0.079$$

for algorithm AVOID_CONTENTION and

$$d_{CAB}(\eta) \approx 0.569 \cdot \eta + 0.021$$

for algorithm COMPARE_AND_BALANCE. Comparing these expressions we see that algorithm AVOID_CONTENTION is even slightly less sensitive to uncertainty than algorithm COMPARE_AND_BALANCE is.

Our simulations suggest that, in contrast to observing cost with uncertainty, observing load with uncertainty does not have any influence on the behavior of the algorithms at all. After having stressed the importance of sampling channels proportionally to their load, this is, at first sight, surprising.



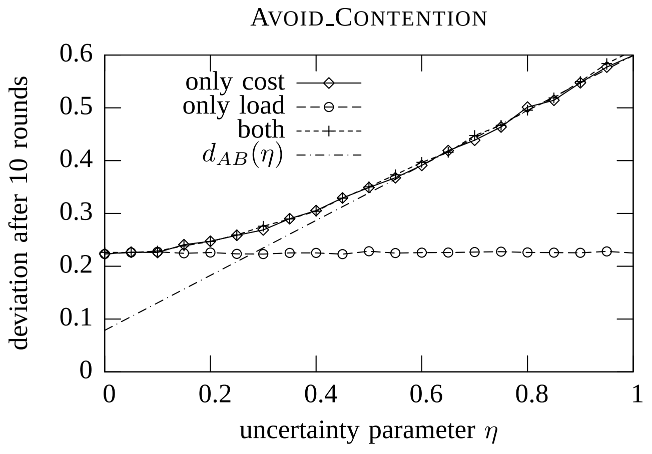


Fig. 3. Relative standard deviation after 10 rounds if cost and/or load are observed under the influence of error.

There is, however, a simple explanation for this. Though observing the value of n_i with uncertainty adds an additional amount of randomness to our algorithm, the expected number of agents that sample a particular channel does not change due to this effect, since underestimating n_i is as probable as overestimating it by the same amount.

We consider the results of these simulations in a simplified model to be very encouraging for the implementation of load-adaptive channel allocation algorithms. Note that algorithm AVOID_CONTENTION relies only on the observation of the cost (e. g. bandwidth) of its own channel. This value can be observed easily without any uncertainty. It is harder to obtain estimates for the values of n_i for the other channels, but we have seen that a very coarse estimate suffices as long as underestimation is as probable as overestimation.

VII. CONCLUSIONS

We have studied in this paper the secondary user channel allocation subject to selfish load balancing game-theoretical approach, and designed two algorithms COMPARE_AND_BALANCE and AVOID_CONTENTION to reach an equilibrium solution. We have shown that the algorithms converge very fast and that the uncertainty of the channel load in the sampling process does not have any severe influence on the performance of the algorithms. Furthermore the results show that the convergence behaviour of the algorithm is independent of tested utility function type. The presented system is admittedly a simplified one, but we have made it simple but realistic enough to underline the algorithm designs and to analyze their convergence properties. We consider the simulation results very encouraging for the implementation of a load balancing adaptive channel assignment solution and testing it to a more complex system model.

ACKNOWLEDGEMENT

The authors acknowledge the financial support of DFG through German UMIC-excellence cluster at the RWTH Aachen.

REFERENCES

[1] J. Mitola, *Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio*. Ph.D. Thesis, KTH, 2000.

[2] M. Duque-Anton, D. Kunz, and B. Ruber, "Channel assignment for cellular radio using simulated annealing," *IEEE Trans. on Vehicular Technology*, vol. 42, pp. 13–21, 1993.

[3] N. Funabiki and Y. Takefuji, "A neural network parallel algorithm for channel assignment problems in cellular radio networks," *IEEE Trans. on Vehicular Technology*, vol. 41, pp. 89–96, 1988.

[4] L. Narayanan, "Channel assignment and graph multicoloring," *Handbook of wireless networks and mobile computing*, pp. 71–94, 2002.

[5] W. Wang and C. K. Rushforth, "An adaptive local-search algorithm for the channel-assignment problem (CAP)," *IEEE Trans. on Vehicular Technology*, vol. 45, pp. 459–466, August 1996.

[6] K. Aardal et al., "Model and solution techniques for frequency assignment problems," *Technical report*, Dezember 2001.

[7] J. Riihijärvi et al., "Performance evaluation of automatic channel assignment mechanism for IEEE 802.11 based on graph colouring," in *Proceedings of IEEE PIMRC, Helsinki, September*, 2006.

[8] K. Leung and B. Kim, "Frequency assignment for IEEE 802.11 wireless networks," *Proc. of IEEE 58th Vech. Tech. Conference Vehicular Technology Conference*, vol. 3, pp. 1422–1426.

[9] F. Gamba, J.-F. Wagen, and D. Rossier, "A simple agent-based framework for adaptive wlan frequency management." *Proc. of MOBICOM*, September 2003.

[10] L. Berlemann et al., "Spectrum load smoothing for cognitive medium access in open spectrum," in *Proc. of IEEE PIMRC*, vol. 3, 2005, pp. 1951–1956.

[11] A. Laufner and A. Leshem, "Distributed coordination of spectrum and the prisoner's dilemma," in *Proc. IEEE DySPAN*, vol. 1, 2005, pp. 94–100.

[12] S. H. Wong and I. J. Wassell, "Application of game theory for distributed dynamic channel allocation," *Proc. of IEEE Trans. on Vehicular Technology Conference, Birmingham, AL, USA*, vol. 1, pp. 404–408, 2002.

[13] P. Mähönen and M. Petrova, "Flash crowds in cognitive radio environment: Beware an unpredictable mob," *Submitted to PIMRC 2007 (available as a technical report).*

[14] R. W. Rosenthal, "A class of games possessing pure-strategy Nash equilibria," *Int. Journal of Game Theory*, vol. 2, pp. 65–67, 1973.

[15] D. Fotakis et al., "The structure and complexity of Nash equilibria for a selfish routing game," in *Proc. 29th Int. EATCS Coll. on Automata, Languages and Programming (ICALP)*, Malaga, Spain, 2002, pp. 123–134.

[16] A. Tanenbaum, *Computer Networks*, 4th ed. Pearson, 2003.

[17] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, March 2000.

[18] E. Even-Dar and Y. Mansour, "Fast convergence of selfish rerouting," in *Proc. 16th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA)*, 2005, pp. 772–781.

[19] S. Fischer and B. Vöcking, “On the evolution of selfish routing,” in *Proc. 12th Ann. European Symp. on Algorithms (ESA)*, ser. Lecture Notes in Comput. Sci., S. Albers and T. Radzik, Eds., no. 3221. Bergen, Norway: Springer-Verlag, September 2004, pp. 323–334.

[20] S. Fischer, H. Räcke, and B. Vöcking, “Fast convergence to Wardrop equilibria by adaptive sampling methods,” in *Proc. 38th Ann. ACM. Symp. on Theory of Comput. (STOC)*. Seattle, WA, USA: ACM, May 2006, pp. 653–662.

[21] S. Fischer and B. Vöcking, “Adaptive routing with stale information,” in *Proc. 24th Ann. ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing (PODC)*, M. K. Aguilera and J. Aspnes, Eds. Las Vegas, NV, USA: ACM, July 2005, pp. 276–283.

[22] P. Berenbrink, T. Friedetzky, L. A. Goldberg, P. Goldberg, Z. Hu, and R. Martin, “Distributed selfish load balancing,” in *Proc. 17th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA)*, 2006.