In IEEE, editor, Proc. of IEEE Wireless Communications and Networking Conf., Atlanta, Georgia, USA, March 2004

Regionalizing Global Optimization Algorithms to Improve the Operation of Large Ad Hoc Networks

D. Hollos, H. Karl, A. Wolisz
Telecommunication Networks Group
Technical University of Berlin
hollos|karl|wolisz@ee.tu-berlin.de

Abstract— **When optimizing operations of large wireless ad hoc networks, neither global nor local information-based approaches fit well: they require either information about the entire network structure, which is in most cases not possible to get, or are not capable of optimizing beyond a very narrow horizon. We propose a novel optimization scheme based on regional information, to compute network-wide optimizations taking the peculiarities of large ad hoc networks into account, obtaining an "emergent algorithm" out of a global optimization algorithm. Our solution uses a clustering algorithm to define regions but needs neither cluster maintenance nor inter-cluster communication protocols, thus is expected to be very robust. The problem of distributed frequency assignment is used as a case study to demonstrate the performance of our method, compared to algorithms based on local- or global information.**

I. Introduction

With the wide-spread deployment of wireless LANs, problems appear that were absent when wireless access points (APs) were still only scarcely deployed. For example, in an office or apartment building where wireless APs are setup independently by many different users, the problem of selecting the operating frequency for each AP becomes challenging – only a small number of frequencies have to be distributed over many, potentially relatively close-by APs to ensure an appropriate spatial reuse of the scarce wireless spectrum. This is akin to the frequency assignment problem commonly known from wide-area cellular systems like GSM, but it is different due to the absence of a global entity or management system that distributes and assigns spectrum to each access point – access points belonging to different companies in an office building are unlikely to have a joint management that assigns frequencies to them. Hence, an automated, distributed solution is necessary that assigns channels to APs without manual information collection (e.g., which AP is placed where and interferes with which other one) and decision making. To support simple, ad hoc deployment, ideally components of the system should only communicate wirelessly with their immediate neighbors which also limits far-range exchange of information as this would be complicated by the ensuing multi-hop nature of the network. This problem is aggravated by the, compared to GSM-type systems, much higher dynamics and possibly even mobility of the entities in such a setup, rendering any centralized, management-protocol-based solution quite impossible.

In fact, this problem of assigning frequencies to APs is but an example of a more general problem class: Given a graph consisting of nodes and edges, where the nodes are attributed by some parameters, and a global target function, find a parameter setting for each node that optimizes the target function; edges can express local constraints on the possible settings of these parameters. For the frequency allocation example, nodes correspond to APs, parameters to the assigned channel, edges express that connected nodes must not be assigned the same channel to avoid interference, and the global target function could be to minimize the number of channels required to solve the assignment problem (effectively, a graph coloring problem). The challenge to solve such an optimization problem here is the absence of a global entity or even of global information about this graph: any node only has locally observable information about itself and its neighbors (e.g, the number of interfering nodes) at its disposal. Hence, any global optimization algorithm (e.g., a graph coloring algorithm for the frequency assignment problem) is not applicable to ad hoc scenarios like the one outlined above where nodes cannot obtain global information.

An alternative approach to such an optimization problem would be to have every node act purely based upon local information. The problem here is that the global optimization goal – the value of the target function – may not be observable locally and therefore, purely local algorithms are often not the best possible choice for problems of this class. Rather, algorithms that cooperate with entities beyond just their immediate neighbors often produce superior optimization results – they belong to the group of *emergent algorithms*, which "produce global effects through cooperative local actions distributed throughout a system" [1].

Applying the idea of emergent algorithms to problems like frequency assignment, however, is anything but straightforward; insight from the global optimization algorithms is still helpful in deriving good solutions. We hence propose a framework that allows us to use global optimization algorithms like emergent algorithms, by confining the operation of the global algorithm to *regions*; the framework executes these regional optimizations in parallel and puts into place the necessary cooperative behavior between these parallel instances. The operation of this framework is distributed and executes the regional optimization steps iteratively; the core ideas are to forego any *explicit* communication between regions but rather to rely on implicit information passing between regions over

several iterations. This approach has also the advantage that regions do not have to be maintained explicitly, which holds the promise to make this framework robust in the face of mobility.

To investigate this idea, we first contrast it with related work in Section II, Section III then describes our framework in detail and Section IV evaluates its performance for the distributed frequency assignment case study. Finally, Section V presents the conclusions.

II. Related work

A. Distributed Optimization Algorithms

The problems discussed in this paper belong to the class of Distributed Constraint Satisfaction Problems (DCSP) [9] and Distributed Constraint Optimization Problems (DCOP) [6]. DCSPs satisfy constraints defined over variables in nodes, no matter what the quality of the result is. DCOPs also care about the "optimality" of the result. Both of them cover network resource allocation problems, graph coloring problems, maximum independent set problems, etc. However, the number of iterations and inter-node communication they require are propotional to the number of nodes, thus their application to real problems is limited [4]. An *incomplete* algorithm is also presented in [4]; however, it needs an agent which periodically has a global view of the whole distributed assignment.

A group of algorithms intended to solve DCOPs is called *emergent algorithms*: multiple instances of the same algorithm run in parallel and these instances make cooperative local actions in order to reach a common global goal; this global goal may not be interpreted – thus cannot be measured – locally [1]. Our proposed method belongs to this group. It makes possible to parallelize and conduct optimization in mid-size to very large areas not using any kind of global knowledge, giving quite good results in terms of "optimality". Moreover, it can be generalized easily to various optimization problems.

B. Frequency assignment

Although there is a vast literature on dynamic channel allocation problems (our case study), especially for GSM systems [2, 3, 10], they are not directly applicable to packet-oriented systems like WLANs. This is due to the different requirements of circuit-switched networks where the main metrics are the call dropping and call blocking rate; moreover, the statistics of outgoing call initiation is different from the statistics of packet sources.

There are only a few papers dealing with automated frequency planning for packet-oriented and cell-based systems like IEEE 802.11 or ETSI HiperLAN/2 [5, 7]. A solution based on *central* knowledge of the network topology is presented in [5]. The paper formulates the problem of allocating frequencies to IEEE 802.11 multi-cells as a 0-1 programming problem where the optimal solution is shown to be NP-complete. A heuristic is also presented which resulted in very good suboptimal solutions. A method partially based on central knowledge is presented in [7]. A Central Station (CS) collects interference measurement results from the Access Points (AP) belonging to it, and periodically creates ordered lists of available frequency channels dedicated to each AP, individually. The APs use these lists to select a channel e.g. when the received interference exceeds a threshold.

As the first algorithm requires central knowledge, its applicability to ad hoc networks is rather limited; the second one has extended possibilities, because it does not use the central information for each decision but for longer periods. In case of ad hoc networks e.g. with *non-limited* size, however, not any kind of global knowledge is available.

III. The Intermittent Cluster Method

A. Overview

The purpose of the proposed framework is to optimize the network by dividing it into several regions and execute several instances of a preselected optimization algorithm "plug-in", which is usually applied in a global manner. These instances run in parallel, one for each region, such that they are unaware of each other and are restricted to their regions. Because of this isolation of these instances (which is necessary as the "plug-in" algorithms are not prepared to cooperate with other instances, being global algorithms in nature), it is unlikely that a single such iteration will suffice to optimize the global target function. Hence, several iterations of such parallel instances will usually be necessary.

The framework takes care to orchestrate the execution of these instances such that a cooperative behavior is achieved despite the isolation of the individual executions. To do so, the framework needs three main components: (i) Deciding when and where a region is created, (ii) the optimization algorithm, which is regarded as a black box "plug in" into the overall framework, and (iii) arranging the communication within a region to provide the necessary information to the global optimization algorithm and to distribute its decisions.

On a high level, the framework proceeds in five steps (see Figure 1): Based on some rule (the *Distributed Leader Selection Method*, DLSM), the decision to create a cluster is taken; the node that takes this decision will be the *leader* of a cluster (step 1). This leader starts to grow a cluster around itself, including all nodes up to a maximum hop count or until it hits another cluster grown by another leader (step 2). Within this cluster, information relevant for the optimization process is collected (e.g., local interference) and sent to the leader, where the optimization plug-in is executed for this cluster (steps 3 and 4). Finally, the computed new parameters are distributed to all members of the cluster (e.g., new frequencies for every AP) in step 5, and the cluster is immediately dissolved Then, the process repeats by (possibly) deciding to create new clusters, usually elsewhere (typically centered near to the borders of the previous regions, see below).

The crucial aspects of this framework are the rules to select leaders (the DLSM) and how to create a cluster once a leader has been determined. Once a leader has been selected and the cluster constructed, collecting the input data for the optimization algorithm is a simple convergecast, distributing

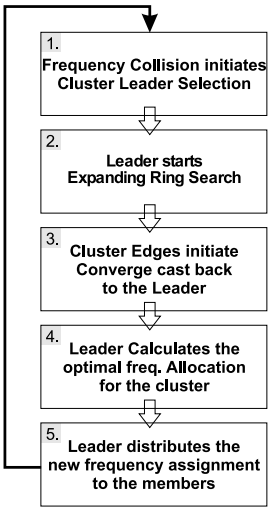


Fig. 1. Steps of one iteration process

the results to all members of a cluster is a simple flooding within the cluster. The particular optimization algorithm is a black box anyway and hence out of scope of the present discussion[1].

B. Growing clusters around a leader

Growing a cluster is done by a simple expanding ring search, starting from the leader, constrained by a *maximal cluster radius* (MCR) as the largest hop count that a cluster member can be away from its leader.

This growing process becomes complicated if two leaders are close to each other (less than 2MCR apart) such that there are entities which could belong to either cluster, and where the clusters touch during their growth. In such a case, the touching nodes do not continue the growth of the clusters; these nodes on the edge shall belong to *both* (or even more) clusters, and they are called *common edge nodes* or simply *common edges*. Figure 2 illustrates this behavior.

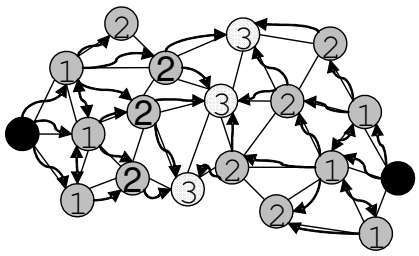


Fig. 2. Example of "touching" clusters: Common Edges are the nodes which are three hops away from either leader (dark nodes)

Common edge nodes have a specific behavior since they could receive potentially conflicting new parameter settings from two or more different leaders. To prevent this, their original values are given as fixed constraints to the optimization algorithms running in the leaders: the values of the common edges are not changed during that particluar iteration but taken into account by the optimization plug-ins. Thus, while a single optimization step can remove constraint violations within a cluster, this restriction has the consequence that any potential violations at or across such a common edge will likely not be resolved.

[1]For the example in Section IV, a concrete graph coloring algorithm will be chosen

C. Distributed Leader Selection Method

The most challenging aspect of this framework is to determine where and when a region should exist and the corresponding cluster should be grown. The goal is to optimize the global target function as quickly as possible, or equivalently, with as few iterations as necessary.

Selecting the leader of a cluster can be triggered by two causes: First, an entity could observe some local constraint violations; second, information from a previous iteration can trigger leader selection.

Selection based on local constraint violation. If one (or more) entity observes local constraint violations (e.g. in the case study, a frequency collision with a neighboring access point), the actions of this entity depend on whether it is currently or has recently (within a time span of $\mathrm{MCR} \cdot T_{\mathrm{hop}}$, where T_{hop} is the maximum time to communicate over one hop) been part of a cluster – if not, such an entity is called *idle*.

An idle entity probabilistically decides whether or not to become a leader. Assuming that it detects constraint violations with N neighbors, it will become leader with probability $P_{\,\mathrm{leader}} = \frac{1}{N+1}$ (assuming uniform distribution of terminals; other distributions would require an adopted rule, which is not in the scope of this paper). This is a heuristic to reduce the number of neighboring entities that start to grow a cluster, lest a large number of tiny clusters in immediate neighborhood would be created, which would be useless.

A non-idle entity detecting constraint violations can rely upon the fact that there is an optimization process in progress and will reach it within maximum $\mathrm{MCR} \cdot T_{\mathrm{hop}}$.

Selection based on information from previous iterations. At the end of an iteration (Section III-B), the most likely place for remaining constraint violations are the borders of regions. Hence, it is likely a good choice to place new leaders on these region borders to start new clusters from there to remove these remaining violations.

To implement this heuristic, an implicit cooperation between common edge nodes takes places via random numbers: Any common edge node chooses a random number and transmits this number to all of its leaders (during the convergecast, in Phase 3 of Figure 1). Each leader selects, in Phase 4, the largest such random number it has received and sends it back to all of its common edge nodes in Phase 5. A common edge node that receives its own random number back from even only a single of its leaders will become a new leader in the next iteration – of course, only if this new prospective leader

actually has observed any constraint violations. This procedure is illustrated by the four clusters in Figure 3, where the common edge node with random number 300 will be the only new leader for the next iteration. A future extension will be to fine-tune this decision based on topology information obtained in previous phases (as common edges receive information from multiple clusters, see also Section III-D).

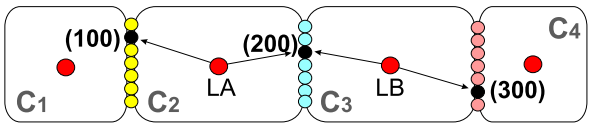


Fig. 3. Distributed leader selection based on random numbers generated in common edge nodes

The result of this process is that new clusters will indeed by formed centered around the former borders, but also a needless explosion of leaders is avoided by "sparsing" the set of possible new leaders appropriately; this "sparsing" effect helps in overcoming situations where the density of leaders is initially high.

D. Discussion

This iterative execution of regionally restricted "global" optimization algorithms has a number of appealing properties. First, the process is triggered on-demand by observing some local optimization need but can be continued, by means of the common edge mechanism, even if no locally observable optimization need exists any longer – resulting in the optimization of a global target function which is not locally observable but can be measured within a specific region. Second, this is enabled by the fact that these regions and the clusters that implement them overlap in space, but not in time: entities are covered iteratively by consecutive clusters, allowing them to pass information from one iteration to another (Figure 4). Third, this overlap allows us to implement information exchange between iterations implicitly and to do without any explicit inter-cluster communication that is necessary for other, similar mechanisms. Fourth, the fact that we pass information implicitly between iterations also makes any cluster maintenance unnecessary – clusters only live for a single iteration.

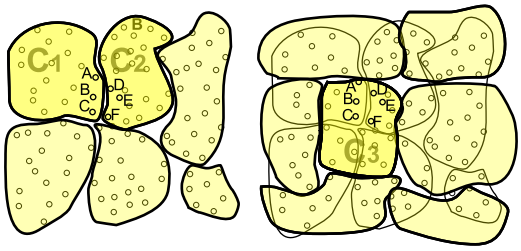


Fig. 4. Two iterations of the framework; new cluster C_3 created on the border between the former clusters C_1 and C_2 (clusters from previous iteration outlined in the right figure)

IV. Performance results

A. Network Structure

We demonstrate the performance using the frequency allocation problem as a case study. Our model consists of cells, each having one randomly placed representative node (AP). All nodes have identical radio range. Two APs are able to communicate with each other through "relayer" terminals if there is one in the radio range of both of them; relayer terminals are also randomly generated. Cells are considered to be neighbors – and thus should use different frequencies – if their APs are able to communicate with each other. The goal of the channel allocation algorithm is to assign frequencies to the cells such that no two neighboring cells get the same frequency. The global optimization goal meanwhile is to reduce the total number of frequencies used.

B. Three algorithms

The channel allocation problem described above was solved by three algorithms for all topologies: a fully distributed method, a central knowledge-based scheme, and our regional scheme.

Fully Distributed Frequency Allocation (DFA). The first basis of comparison is a Random Distributed Channel Assignment Algorithm (RND DCA): Initially the APs have an allocated frequency range (FR); they are allowed to pick their operating frequency from that range autonomously. First, each AP checks whether it observes co-channel interference from neighboring cells using its currently assigned frequency. If so, it chooses a new frequency randomly out of a pool of available, non-interfering frequencies being in its given range. All APs repeat this procedure as long as they experience co-channel interference, or a timeout expires.

If an AP has interference after timeout, it notifies all the others to repeat the channel allocation procedure with increased FR. If the allocation procedure was successful (no such request to increase the frequency range was received) the procedure is repeated with reduced FR; in order to find the lowest number of frequencies needed increasing and decreasing the frequency range is implemented according to the binary search algorithm.

We use this algorithma (see also in [7, 8]) as a basic reference for the number of frequencies.

Global Knowledge-Based Frequency Allocation (GFA). The second basis of comparison is a channel allocation based on global knowledge. In order to make the results easier to compare we used the same greedy-based graph coloring algorithm for both the global and regional optimization. In the case of global knowledge the full network topology was given as input. The algorithm works as follows: first a randomly selected node is assigned the frequency with the lowest index, then the other nodes get always the lowest index possible, which has no direct interference; processing the nodes proceeds in breadth-first search order.

Our Regional Frequency Allocation Scheme. After creating the cluster and collecting the neighborhood-related data,

the leader has the topology graph of its cluster. To assign frequencies, the same greedy graph coloring algorithm is executed in the leader like in the global knowledge-based method.

C. Simulation Scenario

We conducted several simulations with 1500 randomly placed entities (including also the relayer terminals) on an area of 3000 m by 3000 m; the range of a terminal was set to 150 m; we abstracted away details of the wireless communication (MAC, errors, etc.). Five different AP densities were simulated, which resulted in on average 35, 21, 15, 11, and 9 cells overlapping per m^2; the average numbers of neighbors of a cell is 9.5, 5.9, 4.3, 3.3, and 2.9, respectively. The MCR was set to four in order to prevent the simulation area from being covered by a low number of clusters. We use the average number of neighbors when referring to the network density further in the graphs. As the location of APs are selected randomly, nothing prevented the cells from being multiple times overlapped, representing complex structures, e.g. in office buildings.

For each cell density, 30 different random topologies were simulated, each using all three protocols. The initial allocation is chosen such that approximately 90% of the CCs collided with at least one of their neighbors (such bad situation may occur e.g. when switching the devices on).

D. Results

Number of required frequencies. This is the global optimization objective. Figure 5 shows the percentage of channels required by regional optimization and DFA in excess of GFA. The regional results are impressive, as it uses only 7.6% more frequencies on average than GFA, the solution based on global knowledge; moreover, it is practically independent of the network density. In contrast, DFA uses much more frequencies (40.2% on average) and depends on the network density. DFA, however, is faster than regional optimization for cases when the offered range of available frequencies (FR) is large.

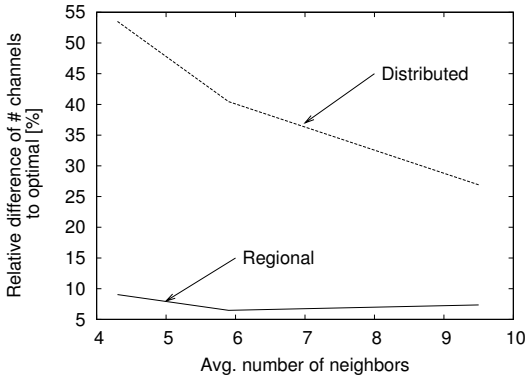


Fig. 5. Relative amount of additional frequency channels of the regional and distributed approach compared to the global allocation

Number of directly colliding cells vs. simulation time. Figure 6 shows how regional optimization converges towards the collision-free solution. The speed of convergence is quite high until the number of collisions gets small, but rather low afterwards – it takes a relatively long time to get rid of the last few collisions. The reason is that although the current implementation of DLSM reduces the number of directly neighboring leaders, such situations still occur and the frequency collision between these two leaders is not resolved. If there are many frequency collisions in the area of the clusters with those neighboring leaders, still numerous of them will be resolved; on the other hand when e.g. there are no more colliding entities in the area of the two clusters, the number of frequency collisions will be unchanged.

The results show that the regional optimization reduces the number of colliding entities quite efficiently up to $\sim 10\%$ (see also Figure 7(b)). This, however, points out to the need for developing DLSM such that it prevents neighboring entities selected to be leaders (see Section V).

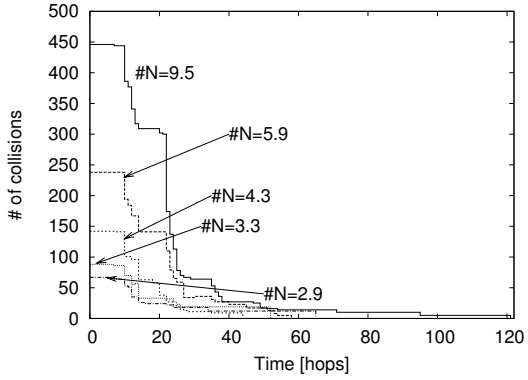
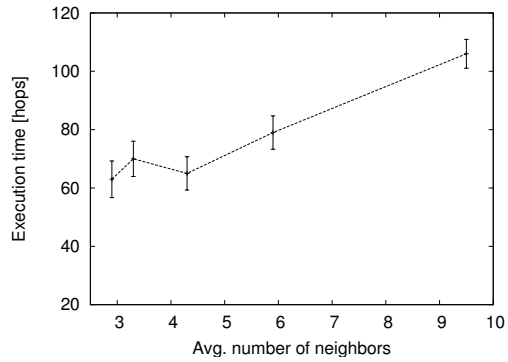


Fig. 6. Number of collisions during execution of the regional algorithm

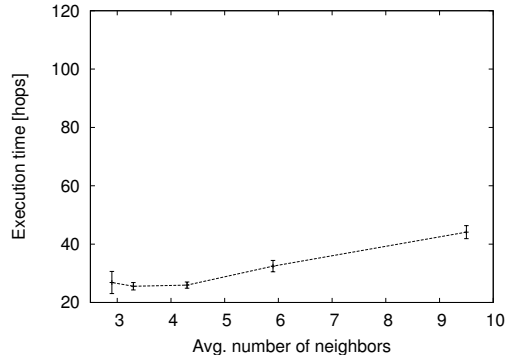
Execution time vs. network density. Figure 7 shows the time (in terms of hops) needed for regional optimization. For Figure 7(a) the termination criterion was that all cells independently decide that there are no constraint violations any longer. The results here look somewhat chaotic, this is due to the non-perfect leader selection of DLSM.

Figure 7(b) shows the execution times when the simulation was terminated as the number of frequency collisions had fallen below 10% of the total number of cells in the network. Although this is a theoretical result (the nodes can not know when the total number of collisions reach 10%) it demonstrates well the potential.

Generated network traffic. The curve in Figure 8 shows how many packets our regional scheme has transmitted per AP on average, versus the average number of neighboring cells. The values indicate that this number scales linearly with the average number of neighbors (which is proportional to the complexity of the frequency assignment problem). Simulations have also shown that – as expected from a parallelized algorithm – the network size neither has an effect on the execution time nor on the average number of packets generated



(a) All frequency collisions resolved



(b) Up to 10% collisions left

Fig. 7. Execution time of regional optimization over average number of neighboring cells relative to the entire population (confidence intervals for 95% confidence level)

per node, if $D_{\text{network}} >> 2\text{MCR}$.

V. Conclusion and Future Work

We have shown that regional optimization is applicable to improve operation parameters of large ad hoc networks. The basic observation is that local optimizations are insufficient, require too much communication overhead or need specialized mathematical analysis; large-area optimizations are too expensive or even impossible. A regional solution is presented which creates clusters overlapping in space but not in time, allowing to carry information from one optimization round to another one. This is done without inter-cluster communication or cluster maintenance.

We demonstrated the benefits of this approach using the frequency assignment problem as a case study. Other global algorithms should be easy to apply simply by replacing the "plug-in" to be executed in the leaders.

The next step is to improve the leader selection to prevent neighboring entities from becoming leaders. In addition, the impact of real mobility and wireless-specific problems (e.g. transmission errors) have to be investigated. A more sophisticated study is necessary about how and when it is possible to bound the convergence times analytically. The results of such a study should further improve the leader selection algorithm. We also intend to give a precise mathematical definition and constraints which "plug-in" functions must meet to be applicable to our framework.

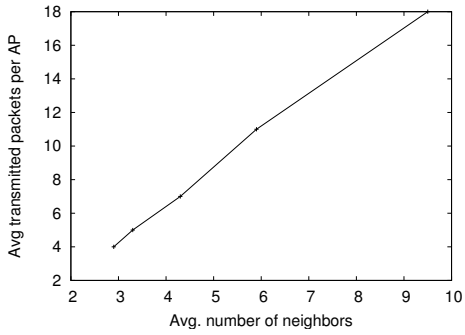


Fig. 8. Average number of packets transmitted by each CC

References

[1] David A. Fisher and Howard F. Lipson. Emergent Algorithms for Survivable Systems. In *Information Survivability Workshop*. Carnegie Mellon University, Software Engineering Institute, October 1998.

[2] D. Grace, T. C. Tozer, and A. G. Burr. Simulation of a Wireless Communications Network Which Employs Distributed Dynamic Channel Assignment. In *Proc. IEE Intl. Conf. on Simulation*. University of York, October 1998.

[3] I. Katzela and M. Naghshineh. Channel Assignment Schemes for Cellular Mobile Telecommunications: A Comprehensive Survey. *IEEE Personal Communications*, pages 10–31, 1996.

[4] Michel Lemaitre and Gerard Verfaillie. An Incomplete Method for Solving Distributed Valued Constraint Satisfaction Problems. In *Proc. of the AAAI Workshop on Constraints and Agents*, 1997.

[5] Kin K. Leung and Byoung-Jo Kim. Frequency Assignment for Multi-Cell IEEE 802.11 Wireless Networks. In *The 57th IEEE Semiannual Vehicular Technology Conf.* IEEE, april 2003.

[6] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. An asynchronous complete method for distributed constraint optimization. In *Proc. of the second Intl. joint Conf. on Autonomous agents and multiagent systems*, pages 161–168. ACM Press, 2003.

[7] Shin Horng Wong. Channel Allocation for Broadband Fixed Wireless Access. In *5th IEEE Intl. Symposium on Wireless Personal Multimedia Communications (WPMC)*. IEEE, oct 2002.

[8] Shin Horng Wong and Ian J. Wassell. Application of Game Theory for Distributed Dynamic Channel Allocation. In *IEEE 55th Vehicular Technology Conf.*, pages 404–408. IEEE, 2002.

[9] Makoto Yokoo, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabara. The Distributed Constraint Satisfaction Problem: Formalization and Algorithms. *Knowledge and Data Engineering*, 10(5):673–685, 1998.

[10] Zhi Hua Zheng and W. H. Lam. Capacity analysis for mobile cellular systems with Distributed Dynamic Channel Assignment. In *Intl. Symposium on Consumer Electronics*, dec 2000.