# Channel Allocation for Dynamic Spectrum Access Cognitive Networks using Localized Island Genetic Algorithm

*(Extended Abstract)*

Mustafa Y. ElNainay, Feng (Andrew) Ge, Ying Wang, Amr E. Hilal, Yongsheng (Sam) Shi,
Allen B. MacKenzie, and Charles W. Bostian
Bradley Department of Electrical and Computer Engineering, Virginia Tech
Blacksburg, Virginia 24061-0111
Email: {ymustafa, gef, ywang06, amr.hilal, shiys, mackenab, bostian}@vt.edu

***Abstract*—In our demonstration, we show how cognitive radios can be organized to form a cooperative ad-hoc cognitive radio network that utilizes the available spectrum opportunistically and efficiently through channel allocation while at the same time avoiding causing interference to primary users as they become active. Our cognitive radios are Software Defined Radios consisting of a Linux laptop and a Universal Software Radio Peripheral integrated with our extended implementation of the GNU Radio software package and our Localized island Genetic Algorithm implementation for the channel allocation.**

## I. Introduction

One promising technology for dynamic spectrum access is Cognitive Radio (CR) [1] with its ability to observe the surrounding network environment and reconfigure itself to adapt to network changes. However, in order to achieve a network-wide goal like the efficient spectrum utilization, CRs need to be organized together in a way that allow for cooperation and distributed reasoning. A Cognitive Radio Network allows for such cooperation by integrating distributed reasoning throughout the network.

In this demonstration, we show how CRs can be organized to form a cooperative ad-hoc cognitive radio network that utilizes the available spectrum opportunistically and efficiently. We achieve this goal through new dynamic channel allocation algorithm using a localized version of the island Genetic Algorithm (LiGA). Moreover, we have developed signal detection and classification system to detect primary users as they become active and avoid causing interference to primary users. The following sections introduce the island genetic algorithm and our system architecture details.

## II. Island Genetic Algorithm

Genetic algorithms, in general, are population-based algorithms that incorporate random methods to search spaces that contain many local maxima and are too large for a complete search. An island Genetic Algorithm (iGA) is a computationally distributed version of the genetic algorithm that divides the population into subpopulations, or islands, that interact through the migration of individuals to other islands [2]. This migration is performed according to a migration policy, which defines where and when individuals move.

The iGA, in general, uses global information. Each node uses the same length and structure for individuals, which in our channel allocation problem would hold information on all communication links in the network. Thus, the iGA distributes only the computation over all nodes, hoping to reach to a good solution faster by letting every node search the whole solution space simultaneously and exchanging best individuals regularly according to a migration policy. In our work, we have modified the iGA in order to use local information instead of global information. The LiGA not only distributes computation over network nodes as with the standard iGA but also limits the information each node uses and shares with other nodes during operation. With the LiGA, each node works in solving a part of the problem and interacts with other nodes to converge to a network-wide solution. Each node now has to operate and keep information on outgoing communication links that originate from all nodes in its interference range instead of all nodes of the whole network.

## III. System Overview

Fig. 1 shows the node architecture for our demonstration. The following sections briefly describe each of the architecture components.

### A. Radio

To design a fully reconfigurable Physical/MAC layer, we need a Software Defined Radio (SDR) development platform. We have chosen GNU Radio as our SDR software architecture and the Universal Software Radio Peripheral (USRP) device as our SDR hardware platform. GNU Radio [3] is an open source toolkit for building software radios. The USRP is an openly designed low-price SDR hardware platform which implements radio front-end functionality and A/D and D/A conversion and uses the Universal Serial Bus (USB2) to connect to the PC that hosts the device. The USRP is fully supported by the GNU Radio library. In our demonstration, we will use two USRPs

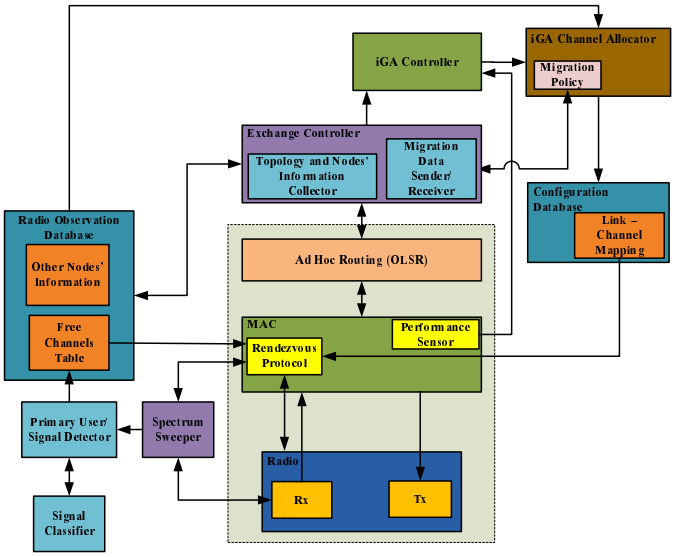


Fig. 1. Node Architecture

at each network node: one is for spectrum detection; the other is for data communication.

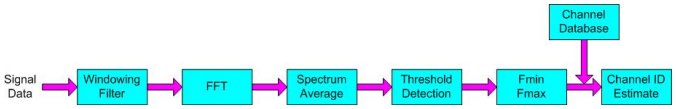


Fig. 2. A Power Spectral Density (PSD) Sensor

### B. Spectrum Sweeper - Signal Detector

Our spectrum sensor, relying on a USRP as the RF front end, is a power spectral density (PSD) sensor. As shown in Fig. 2, it is a Fast Fourier Transform (FFT)-based energy detector. It uses GNU Radio and a USRP to collect baseband signal samples, applies a windowing filter, and then calculates the FFT. The resulting discrete power spectrum density is windowed again to reduce noise irregularity. This averaged spectral data, along with a user specified minimum threshold are used for energy detection. Any spectral energy above the threshold is considered active spectrum. Spectrum with energy below the defined threshold is considered whitespace. Energy detection is used because of two reasons: fast timing restriction and lack of knowledge of the incoming signal.

### C. Signal Classifier

If a signal is detected, our signal classification algorithm shown in Fig. 3 is used to classify the received signal and decide whether it is the primary user or not. The Analog-Digital classification module is used to categorize the signal into one of the following: analog AM, analog FM, or digital modulation. Based on this categorization the next module, bandwidth estimation, will provide a more accurate result. The bandwidth estimated in this module will provide enough information for AM and FM demodulation, before the audio signal is recovered.

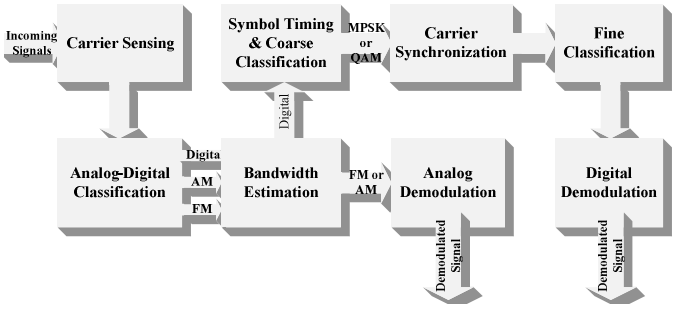


Fig. 3. Signal Classifier

If the received signal is a digital signal, the data will be fed into the symbol timing and coarse classification module. In this module, symbol rate is estimated and signal is sampled at the peak of each symbol; meanwhile, the signal is also classified as QAM or MPSK. This module also tells the order of QAM, for example: 16QAM, or 64QAM. Carrier synchronization is used to estimate and compensate the carrier frequency offset and phase offset. Now all the information required for digital demodulation will have been extracted from the incoming signal, and we can finally implement demodulation. Having the knowledge of the features of the primary users, all signals that content these features are considered as primary users. If the signal is classified as a primary user/signal, an entry will be inserted into radio observation database to mark this channel as unavailable, otherwise this channel will be considered free to be used by secondary users.

### D. Observation Database - Radio Database

Databases of our architecture are implemented in MySQL. The radio observation database combines neighboring nodes' spectrum detection results and stores them in a table that includes active primary users signals' carrier frequency, bandwidth, power, modulation, sensor label (to be replaced by sensor location when available), and the time that each signal was observed. Based on this table, each node is able to identify available channels.

### E. MAC Layer

Since our channel allocation algorithm does not guarantee a conflict free assignment, we use a multiple access technique to access the allocated channel. We use a modified implementation of the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) as our MAC protocol. When a node has some data to send, it first checks the link-channel mapping for the allocated channel to the required destination. If there is no assignment (possibly because of node mobility or new node just joined the network), the Rendezvous protocol is used to communicate with the required destination. If there is allocated channel, the node then checks the free channel table to ensure that this channel is still available for use and sends the data using our CSMA/CA implementation. If the channel is not currently available because of a primary user activity, the Rendezvous protocol is used to communicate with the required

destination and this link-channel relation is inserted into the link-channel mapping table.

### F. Ad-hoc Routing Protocol

In this demonstration, we use an open source implementation of Optimized Link State Routing (OLSR) [4] as our routing protocol. OLSR is a proactive link-state routing protocol which uses Hello and Topology Control (TC) messages to discover and then discriminate link state information throughout the ad-hoc network. Individual nodes use this topology information to compute next hop destinations for all nodes in the network. With its underlying MultiPoint Relays (MPRs) technique, routing control messages and migration information of the LiGA can be delivered to all designated destinations with substantial reduction in the communication cost compared to the classical flooding mechanism.

### G. Exchange Controller

The Exchange Controller is responsible for exchanging information (observations/knowledge) with external nodes. The exchange controller uses OLSR-like control messages to relay spectrum information requests and migration information messages from the LiGA to neighboring nodes. The exchange controller receives spectrum information replies from other nodes and uses them to maintain other nodes' spectrum information stored in the radio observation database. Moreover, the exchange controller responds to other nodes' spectrum information requests with spectrum detection results stored in the radio observation database. The exchange controller and all iGA components are coded using C++ language.

### H. Island Genetic Algorithm Controller

The island Genetic Algorithm Controller is responsible for monitoring the system performance and deciding when it is necessary to optimize the current network configuration as well as how the LiGA process should be configured. The controller also communicates with other nodes to trigger their LiGA processes and collects topology and spectrum information changes since last trigger.

### I. Configuration Database - Link-Channel Mapping

The configuration database is used to maintain the final solution of the LiGA. Once the LiGA Channel Allocator finishes its work, the final link-channel mapping is stored in a table in the configuration database. This table is then used as a reference whenever this node wants to communicate with any other node.

### J. Island Genetic Algorithm Channel Allocator

The main task for the Localized island Genetic Algorithm (LiGA) in this demonstration is to search for the best possible link-channel mapping to maximize the utilization of the available spectrum. We emphasize here that we are planning to use the LiGA in a more general way as the distributed reasoning algorithm for Cognitive Networks and that this demonstration is acting as a proof of concept. Our previous works, using different variations of the iGA in [5] and [6], reveal that the iGA is a promising algorithm to solve more communication and network problems.

After the iGA controller decides to trigger another round of the LiGA process of this node, the iGA controller communicates through the exchange controller with other neighboring nodes to trigger their LiGA processes and collects the necessary topology and spectrum information for the LiGA process to work. The LiGA then starts with the last population from the last run as the initial population for the next run (the first run-first population is randomly generated). Using the last population to seed the new run improves the iGA performance in case of few network changes while a diversity technique, similar to the one used in [7], is used to avoid premature convergence.

In our implementation, we use a simple communication-efficient migration policy in which each node shares with its neighbors only the channel assignment of its outgoing links from its best individual, every fixed migration interval. Each node receiving information from any other node will merge this information with all its individuals if the merged information will give better fitness. Otherwise the receiving node will merge it with only part of its population that is proportional to the fitness of the best individual without the merged information to the fitness with the merged information. This forces each node to consider solutions of other nodes even if it will produce lower fitness individuals than are in the current generation. The main advantage of our LiGA over the standard iGA that uses global information is its scalability with the network size without increasing the time complexity.

## IV. Conclusion

In this paper, we have presented architecture for a cognitive node that can cooperate with other network nodes to utilize the available spectrum opportunistically and efficiently through channel allocation. We have then introduced the Localized island Genetic Algorithm as our architecture distributed reasoning algorithm to perform the channel allocation. We have briefly described each of our node architecture components.

## References


[1] J. Mitola III and G. Q. Maguire, "Cognitive radio: Making software radios more personal," *IEEE Personal Communications*, vol. 6, no. 4, pp. 13–18, Aug. 1999.

[2] E. Alba and J. M. Troya, "A survey of parallel distributed genetic algorithms," *Complexity*, vol. 4, no. 4, pp. 31–52, 1999.

[3] GNU Radio. [Online]. Available: http://www.gnu.org/software/gnuradio/index.html

[4] T. Clausen, P. J. (editors), C. Adjih, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L.Viennot, "Optimized link state routing protocol (OLSR)," RFC 3626, Oct. 2003, network Working Group. [Online]. Available: http://ietf.org/rfc/rfc3626.txt

[5] D. H. Friend, M. Y. ElNainay, Y. Shi, and A. B. MacKenzie, "Architecture and performance of an island genetic algorithm-based cognitive network," in *Proc. IEEE CCNC'08*, Las Vegas, NV, 10–12 Jan. 2008, pp. 993–997.

[6] M. Y. ElNainay, D. H. Friend, and A. B. MacKenzie, "Channel allocation & power control for dynamic spectrum cognitive networks using a localized island genetic algorithm," in *Proc. IEEE DySPAN'08*, Chicago, IL, 14–17 Oct. 2008, pp. 1–5.

[7] K. Rasheed and H. Hirsh, "Using case-based learning to improve genetic-algorithm-based design optimization," in *Proc. of the 1997 International Conference on Genetic Algorithms (ICGA'97)*, East Lansing, MI, 19–23 July 1997, pp. 513–520.