Multi-Objective Multicast Routing Optimization in Cognitive Radio Networks

Yu Jie, Ahmed E. Kamal

***Abstract*—In this paper, we study the multicast routing problem in Cognitive Radio Networks (CRNs). We propose a new network modeling method, where we model CRNs using a Multi-rate Multilayer Hyper-Graph (MMHG). Given a multicast session of the MMHG, our goal is to find the multicast routing trees that minimize the worst case end-to-end delay (*delay*), maximize the multicast rate (*rate*) and minimize the number of transmission links (*numOfLinks*) used in the multicast tree. We apply two metaheuristic algorithms (Multi-Objective Ant Colony System optimization algorithm (MOACS) [1] and A Simulated Annealing-Based Multi-objective Optimization Algorithm (AMOSA) [2]) in solving the problem. We also study the scheduling problem of multicast routing trees obtained using the MMHG model. Our simulation results show that within a few seconds, MOACS can find more than 60% of the approximated Pareto Front (APF) in small CRNs, and AMOSA can find approximately 45%. Moreover, the solutions found by MOACS and AMOSA that are not in the APF are within 10% relative distances to solutions in the APF.**

I. INTRODUCTION

With a growing number of wireless spectrum-hungry services, current wireless networks are becoming overcrowded. To increase spectrum utilization, CRNs are proposed to allow unlicensed secondary users (SUs) to dynamically access licensed spectrum bands without interfering with the licensed Primary Users (PUs). Many researchers explored CRNs' capabilities in supporting different wireless applications. Multicast is one of the important service models, as a growing number of applications rely on multicast, such as scheduled audio/video distribution, gaming and social networking, etc.

There is some recent work in the literature which study multicast support in CRNs. Reference [3] developed an optimal power control policy on base station (BS) and an efficient cooperative communication schedule among SUs so that aggregate throughput on all SUs is maximized. In reference [4], a BS can provide service to both PUs and SUs, and a non-linear program was formulated to find the optimal spectrum and power allocation scheme on BS as well as SUs to maximize the total sum rate of SUs. Reference [5] considered multicast scheduling in a cell formed by a Mesh Router and its served Mesh Clients. It studied spectrum localization, cooperative communication as well as network coding schemes to minimize multicast end-to-end delay. Reference [6] formulated the multicast routing problem as a mixed linear program with a multiple set of communication constraints, and an objective of minimizing the required network-wide resources to support multiple multicast sessions. Reference [7] proposed a distributed and on-demand multicast routing protocol COCAST, where all nodes in the multicast group cooperatively detect other signals, and exchange information through piggybacked Join Query and Join Reply messages using a predefined common channel. Nodes select their channels based on exchanged information.

Different from works introduced above, this paper considers multi-objective multicast optimization problem in CRNs. The goal is to develop methods for finding the Pareto Front of multicast routing trees that can reduce the influence of transmission delay as well as switching delay on multicast end-to-end delays, maximize the throughput and minimize the number of used links of the CRN. Pareto Front is a set of non-dominated solutions that strictly dominate all other possible solutions. We use $X_{solu_i} \succ X_{solu_j}$ to denote that multicast tree solution i strictly dominates multicast tree solution j. $X_{solu_i} \succ X_{solu_j}$ if and only if $delay_i \leq delay_j$, $rate_i \geq rate_j$ and $numOfLinks_i \leq numOfLinks_j$ and there exists at least one strict inequality. We apply two metaheuristic algorithms in finding the Pareto Front. Moreover, we study the scheduling problem of the multicast trees obtained through MMHG. The rest of this paper is organized as follows. The system model is introduced in Section II, and our research problem is described in Section III. In Section IV, we show how to use metaheuristic search algorithms to solve the problem. Simulation results are shown in Section V. Section VI concludes the paper.

II. SYSTEM MODEL

In this paper, we assume a CRN operating on TV White Space channels (Television channels 2-51) [8]. Each channel is licensed to a PU. There is a set of SUs in the CRN that can operate on all available channels, but we assume that each SU has one radio only, and can transmit or receive on one channel at a time. Moreover, each SU knows the availability of all channels and the location of all nodes (including all SUs and PUs) in the CRN. The CRN uses an overlay model, such that an SU's transmission can cause interference to an active licensed PU of the same channel if the SU is located within the PU's interference range, and vice versa. We also assume that the CRN is quasi-static, such that the PUs will not change their activities during our computing process.

Under the assumption that all SUs adopt the best modulation and coding scheme, we apply Shannon-Hartley's formula to calculate the maximum achievable rate of a communication session: $C = Wlog_2(1 + \frac{P_t G}{d^{\alpha} N_0 W})$. C is the maximum achievable rate in $bits/second$, W is the channel bandwidth in Hz, P_t is the transmission power in $Watts$, d in meters is

Yu Jie and Ahmed E. Kamal are with Electrical and Computer Engineering Department of Iowa State University, Ames, IA, 50011
Email: {jysarah, kamal}@iastate.edu

the transmission range of the SU transmitter, G is the antenna gain, N_0 is the noise power spectrum density in W/Hz, and α is the path loss exponent which is in the range from 2 to 4. In our paper, we use W as the TV channel bandwidth, 6 MHz, P_t as the maximum transmission power that is allowed by the IEEE 802.11 protocol which is 100 mW, G as 1, N_0 as the thermal noise power spectrum density -174dbm/Hz, and $\alpha = 2$.

The CRN is modeled using a directed hypergraph. Reference [9] proposed the use of multilayer hypergraph in modeling CRNs. In our paper, we use multilayer hypergraph to model a CRN while introducing a significant extension. In a hypergraph, a hyperedge, also referred to here as a supernode, models a transmission process which consists of a transmitter SU set (TSU) and a receiver SU set (RSU), where they operate on the same channel. TSU only contains one SU, and RSU can have multiple SUs due to the wireless broadcast feature. Let d_r denote the transmission range of the TSU in the supernode. Then, the RSU is a set of SUs that are located within d_r from the TSU. The transmission rate C within a supernode is defined by Shannon-Hartley's formula, with d set to the transmission range d_r. The same SU of the TSU may form a different supernode that transmits at a higher rate, $C^{'} > C$, and over a shorter distance, $d_r^{'} < d_r$, such that the new RSU is a set of SUs within a maximum distance $d_r^{'}$ from the TSU, and a subset of the SUs within a distance d_r from the TSU. The cost of the supernodes with transmission rate C is $1\,data\,segment/C$, where 1 data segment is the size of scheduled transmitted data every transmission cycle, as will be explained below. The hypergraph also has a set of dummy supernodes to represent each transmitting SU and receiving SU. If there are n SUs in the CRN, then n transmitter dummy supernodes (TDsupernode) and n receiver dummy supernodes (RDsupernode) are present in the hypergraph. TDsupernodes and RDsupernodes have no rate, no operating channel, no cost and both their TSU and RSU correspond to the same SU. To distinguish them from dummy supernode, we use Csupernode trefers to communication supernode that is neither TDsupernode nor RDsupernode. TDsupernodes and RDsupernodes will always exist in the hypergraph, but one condition must hold for a Csupernode to be present in a hypergraph: its TSU is not located within the interference range of the active PU licensed on the same channel to be used by itself.

The following conditions should be satisfied when considering the existence of a link between two supernodes in the hypergraph (we define TSU_i and RSU_i as the TSU and RSU of supernode i):

(1) A link exists from Csupernode i to Csupernode j if $TSU_j \subseteq RSU_i$ and the active PU of the same channel used by Csupernode i is not located within the interference range of the SU in TSU_j.

(2) A link exists from TDsupernode i to Csupernode j if TSU_j has the same SU as the SU that TDsupernode i represents.

(3) A link exists from Csupernode i to RDsupernode j if $TSU_j \subseteq RSU_i$ and the active PU of the same channel used by Csupernode i is not located within TSU_j's interference range.

(4) No links exist between TDsupernodes and RDsupernodes.

The cost of a link between two Csupernodes is $k * W * |c_i - c_j|$, where c_i and c_j are the channel numbers used by Csupernode i and Csupernode j, respectively, k is the constant $10ms/10MHz$ [10] and W is the TV channel bandwidth 6 MHz. When $c_i \neq c_j$, the link cost is the switching delay of the same SU in TSU of Csupernode j switching from channel c_i to channel c_j. The costs of links from TDsupernodes to Csupernodes, or from Csupernodes to RDsupernodes are 0 seconds.

Figure 1 shows a simple CRN of two channels, 6 SUs and 2 active PUs, one for each channel. The virtual dotted links between SUs show that they can switch between channels. Here we assume that an SU's transmission range $d_{r_1} = 50$ meters with achievable rate R_1, $d_{r_2} = 100$ meters with rate R_2, and $d_{r_3} = 150$ meters with rate R_3, where $R_1 > R_2 > R_3$. Table I shows the distances among all SUs of the CRN shown in Figure 1.

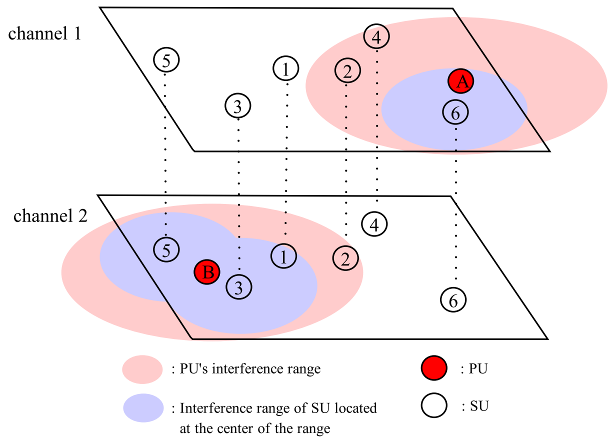
channel 1
5
1
4
2
A
3
6
channel 2
4
5
1
2
B
3
6
: PU's interference range
: PU
: Interference range of SU located at the center of the range
: SU

Figure 1. A multilayer graph representation

Table I
DISTANCES AMONG SUS IN FIGURE 1

Distance(meters)	SU 1	SU 2	SU 3	SU 4	SU 5	SU 6
SU 1	—	50	65	130	180	260
SU 2	50	—	185	55	280	190
SU 3	65	185	—	220	150	320
SU 4	130	55	220	—	330	200
SU 5	180	280	150	330	—	500
SU 6	260	190	320	200	500	—

Using the principles we defined for constructing supernodes and links between supernodes, Figure 2 shows the directed hypergraph mapped from the multilayer graph shown in Figure 1. The six shaded supernodes on the left side of the hypergraph are TDsupernodes, and other six shaded supernodes on the right side of the hypergraph are RDsupernodes. The dashed circles represent Csupernodes operating on channel 1, and the two unshaded undashed circles represent Csupernodes operating on channel 2. Link costs are not shown in Figure 2. We create 3 Csupernodes with TSU as {1}: Csupernodes {1, 2}, {1, 2 3} and {1, 2 3 4} with rates R_1, R_2 and R_3, respectively. The numbers between the curly braces are SUs ID. We also create two Csupernodes with TSU as {3} and one Csupernodes with TSU as {5}. Csupernodes operating on channel 1 with TSU as {2}, {4} or {6} are not present in the hypergraph because SUs 2, 4 and 6 are located within PU A's interference range. On channel 2, we create two Csupernodes with TSU as {4}. Csupernodes with TSU as {1}, {2}, {3} or {5} are not present in the hypergraph, as SUs 1, 2, 3 and 5 are located within PU B's interference range. There are no Csupernodes with TSU as {6} available in the hypergraph since no SUs are located within SU 6's maximum transmission range.

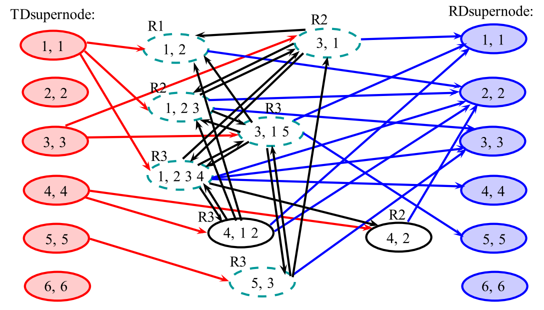

Figure 2. The directed hypergraph mapped from the multilayer graph shown in Figure 1

III. Problem Description

Given a multicast session (s, Des), where s is a source TDsupernode, Des is a set of destination RDsupernodes, our goal is to find the optimal set of multicast routing trees satisfying three objectives: 1) minimal worst case end-to-end delay (*delay*), 2) maximal data rate (*rate*) and 3) minimal number of links (*numOfLinks*) used in the multicast tree. The problem is a multi-objective optimization problem with decision variables $X_{solu} = (x_1, x_2, x_3, ..., x_n)$, where n is the number of all links in the hypergraph, and $x_k \in \{0, 1\}$ is 1 if link k is selected to build the multicast tree, and 0 if not. With the computed Pareto Front of multicast routing trees, we select the final solution out of the Pareto Front depending on which objective is more important.

IV. Multi-Objective Optimization Solvers

Evolutionary Algorithms (EA) are usually used for solving multi-objective optimization problems, Kalyanmoy Deb's Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [11] and Eckart Zitzler's Strength Pareto Evolutionary Algorithm-2 (SPEA2) [12] are two classic EAs which are used for this purpose. The basic idea of NSGA-II is that it builds a population of competing individuals, ranks and sorts each individual according to non-domination level, applies evolutionary operations to create new pool of offsprings, and then combines the parents and offsprings before partitioning the new combined pool into fronts. SPEA2 uses an external archive to contain non-dominated solutions previously found, and non-dominated individuals found through mutation are copied to the external archive at each generation.

Most EAs are suitable for solving optimization problems with continuous decision variables. However, the decision variables in our problem are binary discrete, and are constrained by tree structure. The evolutionary operations to create new pool of offsprings are not efficient for our problem. Therefore, we adopt two other metaheuristic algorithms, which have better features than EAs in solving our problems. The two algorithms are MOACS [1] and AMOSA [2]. For the rest of this section, Subsection A introduces the algorithms for generating a random multicast tree solution. MOACS and AMOSA are described in Subsection B and Subsection C, respectively.

A. Generating a random multicast tree solution

It is not hard to search for a random multicast tree in a given hypergraph. However, we cannot compute its corresponding *delay* and *rate* unless we find its optimal transmission schedule. So in this paper, the process of generating a random multicast tree solution follows four steps:

Step 1, search for a random multicast tree in the hypergraph.

Step 2, modify the tree in order to remove instances of duplicate transmissions. Two Csupernodes i and j are duplicate transmissions when they have the same TSU and $RSU_i \subseteq RSU_j$ or $RSU_j \subseteq RSU_i$. The multicast tree found by Step 1 sometimes has such duplicate transmissions that are wasteful of bandwidth, and should be merged.

Step 3, find the optimal transmission schedule of the modified tree.

Step 4, compute the *delay*, *rate*, and *numOfLinks* of the modified tree.

Step 1 and Step 2 are presented in Subsection 1). Step 3 is explained in Subsection 2). Step 4 is described in Subsection 3).

1) Building a multicast tree: To build a multicast tree in a hypergraph, we apply algorithm BuildTree() from reference [1] with modifications. We start from the source TDsupernode s, and a non-visited supernode is selected at each step. We assign the probability of selecting next un-visited neighbor node of supernode i using Equation (4.1). In the equation, N_i is the set of unvisited neighbor supernodes of supernode i, $C_j = 1/(t_{ij} + t_j)$, t_{ij} is the cost of link (i, j), and t_j is the cost of supernode j. This process continues until all the destination RDsupernodes in Des of the multicast session are reached. In MOACS, the equation to compute the probabilities is different from Equation (4.1), and will be introduced in later sections.

$$p_{ij} = \begin{cases} \frac{C_j}{\sum_{\forall g \in N_i} C_g} & if\, j \in N_i \\ 0 & otherwise \end{cases} \quad \cdots\cdots (4.1)$$

With the multicast tree computed using BuildTree(), we perform following modifications:

(1) Merge supernodes that are duplicate transmissions.

(2) Check destination RDsupernodes's reachability. In the multicast tree build by BuildTree(), some destination RDsupernodes might not be connected to the Csupernodes (whose RSU contains SU corresponding to the RDsupernode) which have the shortest link distance to the source TDsupernode. In reality, this is not the case. We correct such unrealistic connections.

2) Transmission Scheduling: In this subsection, we want to schedule a transmission cycle, which consists of a number of time units that have the same constant duration. There is a set of Csupernodes to be scheduled to transmit at each time unit. Our goal is to compute what Csupernodes of the given multicast tree should be scheduled in which time unit, such that a minimal total number of time units in a transmission cycle is obtained. With the optimal transmission cycle, we compute the time interval needed between each time unit of a cycle, in order to prevent any interference caused by switching between channels.

In the heuristic algorithm, there are two major phases. Phase 1 uses a weighted graph-coloring heuristic algorithm [13] to compute the transmission cycle of a given multicast tree. Phase 2 formulates a simple linear program (LP) to compute the time interval needed between each time unit of a cycle.

In Phase 1, we first build the directed conflict graph F_g of the multicast tree. All Csupernodes of the given multicast tree form all nodes in F_g. For a Csupernode i, we use $IntendDes_i$ to denote the set of SUs that are the intended receivers of TSU_i. The $IntendDes_i$ of TSU_i is the union of TSUs of all supernodes (including RDsupernodes) that have the outgoing link from supernode i in the hypergraph. A link exists from Csupernode i to Csupernode j in F_g when:

(1) Csupernode i and Csupernode j are of the same channel, and TSU_i is located within the interference range of any node in $IntendDes_j$, or $TSU_j \in RSU_i$.

(2) Csupernode i and Csupernode j are on different channels, and $TSU_i \cap TSU_j \neq \emptyset$ or $TSU_j \subseteq IntendDes_i$ or $TSU_i \subseteq IntendDes_j$ or $IntendDes_i \cap IntendDes_j \neq \emptyset$.

To form a weighted conflict graph $F_g^{'}$ from F_g, we assign weight 1 to Csupernodes of the largest rate R_{max} and other Csupernodes are assigned with weight $\lceil R_{max}/R_i \rceil$ where R_i is the rate of the Csupernode i, since the same amount of data transmitted by Csupernode of R_{max} will take more time to be transmitted by Csupernodes with rate lower than R_{max}. Note that we can also use least common multiple (LCM) of rates of all Csupernodes in the multicast tree, divided by R_i to denote the weight of each Csupernode i, but this may cause the cycle length too long to be a feasible transmission schedule.

With the directed conflict graph F_g and weights for all Csupernodes, we apply the weighted graph-coloring algorithm proposed in reference [13] to compute the transmission cycle.

The following gives an example of a computed transmission cycle. In the hypergraph, {55} is the source TDsupernode, s, and {72, 73, 74, 75} is the destination RDsupernode set, Des. BuildTree() generates a random multicast tree which is shown in Figure 3.

Csupernodes 9 and 30 have rate 36.5 Mbits/s. Csupernodes 38 and 35 have rate 52.2 Mbits/s. So $R_{max} = 52.2$ Mbits/s. The weight for Csupernodes 38 and 35 is 1, and weight for Csupernodes 9 and 30 is $\lceil 52.2/36.5 \rceil$, which is 2. The computed transmission cycle $XsoluSche$ of the multicast tree shown in Figure 3 is:

unit 1: 9, 38
unit 2: 9
unit 3: 30, 35
unit 4: 30

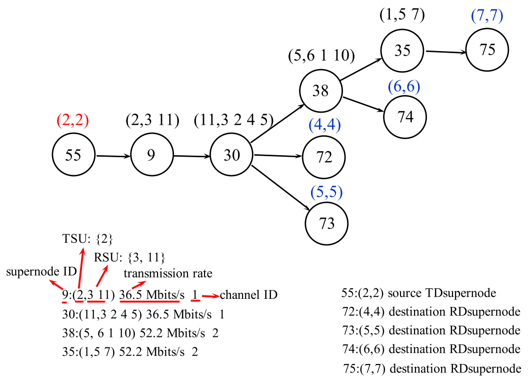
(2,2)
55
(2,3 11)
9
(11,3 2 4 5)
30
(5,6 1 10)
38
(1,5 7)
35
(7,7)
75
(6,6)
74
(4,4)
72
(5,5)
73
TSU: {2}
RSU: {3, 11}
supernode ID
transmission rate
9:(2,3 11) 36.5 Mbits/s 1
channel ID
30:(11,3 2 4 5) 36.5 Mbits/s 1
38:(5, 6 1 10) 52.2 Mbits/s 2
35:(1,5 7) 52.2 Mbits/s 2
55:(2,2) source TDsupernode
72:(4,4) destination RDsupernode
73:(5,5) destination RDsupernode
74:(6,6) destination RDsupernode
75:(7,7) destination RDsupernode

Figure 3. A random multicast tree

In Phase 2, we formulate an LP to compute the time intervals needed between each time unit of a cycle. Csupernodes of different time units may have different operation channels, and the time intervals between each time unit are used for SUs to switch between channels. In the LP, our objective is to minimize the sum of all time intervals of a cycle. With the multicast tree example in Figure 3 and its transmission cycle computed in Phase 1, the LP problem is formulated below:

We use $[inter\, i]$ to represent the value of time interval i in seconds between time units i and $i+1$, use $[unit\, time]$ to represent the period of one time unit in seconds, and $[1\, data\, segment]$ to represent size of the transmitted data segment in bits.

$Minimize: \quad [inter\,1] + [inter\,2] + [inter\,3] + [inter\,4]$
$Subject\ To:$
A: for unit 1
$[inter\,4] \geq 0.006 * |2-1| \cdots\cdots\cdots\cdots (1)$
B: for unit 3
$[inter\,1] + [unit\,time] + [inter\,2] \geq 0.006 * |1-2| \cdots (2)$
C: for each time interval
$[inter\,1] \geq 0 \cdots\cdots\cdots\cdots (3)$
$[inter\,2] \geq 0 \cdots\cdots\cdots\cdots (4)$
$[inter\,3] \geq 0 \cdots\cdots\cdots\cdots (5)$
$[inter\,4] \geq 0 \cdots\cdots\cdots\cdots (6)$
D: for each time unit
$[unit\,time] = [1\,data\,Segment]/R_{max} \cdots\cdots\cdots\cdots (7)$

The constraints above show that there should be sufficient time gap for SUs to switch channels between two successive time units where they are scheduled for operating. In the constraints above, $0.006 = k*6MHz$ and k is $10ms/10MHz$.

This is a polynomial time solvable LP, and we use CPLEX to solve it. Notice that a sufficiently large size of $[1\, data\, segment]$ can result in a large $[unit\, time]$, such that the sum of time intervals in the transmission cycle can be minimized. However, the size of 1 data segment can not be set too large, otherwise the retransmission cost will be huge when network is unstable. A tradeoff needs to be considered here to set the size of 1 data segment, and is not studied in this paper.

3) Computing the objective value: With the generate multicast routing tree, the computed transmission cycle and the computed time intervals between each time unit, we can compute the three objective values of the multicast tree.

$delay$ is the time interval from the time unit where source SU sends the data segment and the time unit where all destination SUs finish receiving the data segment.

$rate$ equals the size of 1 data segment divided by the duration of a transmission cycle, which includes the time intervals between each time unit in the cycle.

$numOfLinks$ equals the number of links of the generate multicast tree.

B. MOACS

In this subsection, we describe the metaheuristic algorithms for computing the Pareto Front of multicast routing trees in a CRN.

Ant colony optimization (ACO) algorithm is a metaheuristic inspired from the foraging behavior of some ant species. The ants leave pheromone at a path they select. The amount of pheromone on a path will cause the ants to eventually choose the shortest path to reach the destination.

Reference [1] proposed the ACO based MOACS to solve multi-objective multicast problem in wired networks. We adopt this algorithm with modifications to solve our problem. Algorithm 1 shows the general procedure of the modified MOACS.

Algorithm 1 General procedure of modified MOACS

```
1: Input: HyperGraph $H(V_H, E_H)$, Multicast Session($s$, $Des$)
2: Output: Optimal multicast tree set $Y_{known}$
3: Initialize $\tau_{ij}$ with $\tau_0$ for all edges,
     numOfIterations = 0, $Y_{known} = \phi$;
4: while(numOfIterations < Iteration Size)
5:     for(i from 0 to popSize)
6:         BuildTree($Xsolu$);
7:         construct transmission schedule of $Xsolu$;
8:         compute delay, rate, numOfLinks of $Xsolu$;
9:         if($Xsolu$ is not dominated by any $Xsolu_k \in$
              $Y_{known}$) then
10:            $Y_{known} = Y_{known} \cup Xsolu -$
               $\{Xsolu_k | Xsolu \succ Xsolu_k\}$,
                 $\forall Xsolu_k \in Y_{known}$
11:        end if
12:        if($Y_{known}$ was modified) then
13:                $\tau_{ij} = \tau_0 \;\; \forall (i,j) \in E_H$
14:        else
15:            repeat $\forall Xsolu_k \in Y_{known}$
16:               $\triangle\tau_{best} = w_1 * delay_k$
                            $+ w_2 * rate_k + w_3 * numOfLinks_k$
17:               $\tau_{ij} = (1-p)\tau_{ij} + p\triangle\tau_{best} \;\; \forall (i,j) \in E_H$
18:            end repeat
19:        end if
20:     end for
21:     numOfIterations++;
22: end while
```

Given the HyperGraph $H(V_H, E_H)$, we first initialize pheromone matrix τ_{ij} with τ_0 on all edges. In each iteration, with each X_{solu} found by an ant, a known Pareto Front Y_{known} is updated including the best non-dominated solutions that have been found so far. If Y_{known} changes, τ_{ij} will be re-initialized to improve the exploration in the decision space. Otherwise, τ_{ij} is updated using the solutions in Y_{known} to better exploit the knowledge of the best known solutions. In BuildTree() of Algorithm 1, we calculate the probabilities of selecting the next un-visited neighbor node using Equation (4.2). In this equation, τ_{ij} is the pheromone of link (i, j), $C_j = 1/(t_{ij} + t_j)$, t_{ij} is the cost of link (i, j), and t_j is the cost of supernode j. By setting α and β, we can adjust the weight of τ_{ij} and C_j in computing the probabilities. A pseudo-random procedure [1] is also used in selecting the next non-visited supernode.

$$p_{ij} = \begin{cases} \frac{\tau_{ij}^{\alpha} C_j^{\beta}}{\sum_{\forall g \in N_i} \tau_{ig}^{\alpha} C_g^{\beta}} & if\, j \in N_i \\ 0 & otherwise \end{cases} \quad \cdots\cdots (4.2)$$

C. AMOSA

AMOSA is an algorithm proposed in reference [2] to use simulated annealing to solve multi-objective optimization problems. We adopt the code provided by reference [2] and modify the process of constructing a random multicast tree solution as introduced in previous subsections. AMOSA will return an archive which records all optimal multicast routing trees that can be found within a given number of iterations.

Notice that AMOSA starts with a set of initial solutions. An archiving process is applied to the initial solution set to select all Pareto optimal solutions within the solution set and put them into an archive. One solution (current solution) is randomly selected from the archive. Then the iteration process starts by generating a new solution from the current solution through mutation. After comparing the new solution with current solution and optimal solutions in the archive found so far, it will decide whether to put the new solution into the archive, drop the new solution or set the new solution as the current solution. The process repeats for a given number of iterations.

V. Simulation Results

In this section, we study the performance of MOACS and AMOSA in finding optimal multicast tree solutions in a hypergraph. Due to the dynamic nature of CRNs, we study the performance of MOACS and AMOSA after running them for a few seconds. We study the following three metrics of MOACS and AMOSA: 1) the percentage of solutions in true Pareto Front found by MOACS and AMOSA; 2) the distances of solutions found by MOACS and AMOSA from solutions in true Pareto Front; 3) running time comparison between MOACS and AMOSA.

To approximate the true Pareto Front, we run a Uniformly Random Search Algorithm (URSA) for millions of iterations. The procedure of URSA is very similar to MOACS, except URSA does not consider the influence of pheromone. On a computer with 31 GB memory and 8 core Intel Xeon CPU E5440@2.83GHz, we run a simulation of 20 randomly generate CRNs. Each network has 20 SUs (with ID from 0 - 19), 3 different channels, and 3 primary users, one for each channel. There are 1 source TDsupernode, 5 destination RDsupernodes for one multicast session in each CRN. For all SUs, we use three different transmission ranges corresponding to three different transmission rates. We set the size of 1 data segment as 1 Mbits. We use Y_{utrue} to denote the approximated true Pareto Front (APF) found using URSA, and Y_{MOACS} and Y_{AMOSA} to denote the solution set found by MOACS and AMOSA, respectively. For these 20 randomly generate CRNs, we run URSA for 1,000,000 iterations to approximate Y_{utrue}, and run MOACS and AMOSA both for 200 iterations and 1000 iterations to compute Y_{MOACS} and Y_{AMOSA}. In Table II, we record sum of solutions in Y_{utrue} of 20 CRNs, sum of solutions in Y_{MOACS} and Y_{AMOSA} that exist in the Y_{utrue} of 20 CRNs, and compute the average percentages of solutions out of Y_{utrue} that Y_{MOACS} and Y_{AMOSA} have and the average relative distances of solutions in Y_{MOACS} and Y_{AMOSA} that are not in Y_{utrue} from solutions in Y_{utrue}.

Table II
PERCENTAGES OF SOLUTIONS IN APF THAT MOACS AND AMOSA FOUND THROUGH 200 AND 1000 ITERATIONS AND AVERAGE RELATIVE DISTANCES FROM APF

Iterations	1,000,000	200	200	1000	1000
Solution Set	Y_{utrue}	Y_{MOACS}	Y_{AMOSA}	Y_{MOACS}	Y_{AMOSA}
Sum of Solutions in APF	62	26	28	39	28
Percent-age of Y_{utrue}	—	41.94%	45.16%	62.9%	45.16%
Relative distance from Y_{utrue}	—	6.7%	8.81%	5.14%	9.1%

Of all 20 simulation CRNs, the optimal solutions in APF that AMOSA finds within 200 iterations are the same as in 1000 iterations, and are an average of 45.16% of the APF. Moreover, the optimal solutions found by AMOSA are mostly from the archiving process of the initial solution set, and the iteration process can not improve its solutions significantly. However, MOACS finds more solutions of the APF by performing more iterations. Within 200 iterations, MOACS finds an average of 41.94% of the APF, and it finds an average of 62.9% of the APF within 1000 iterations. To study the closeness of solutions in APF and solutions found by MOACS and AMOSA, we use $(|delay_i - delay_j|/delay_i + |rate_i - rate_j|/rate_i + |numOfLinks_i - numOfLinks_j|/numOfLinks_i)/3$ to compute the relative distance between two solutions i and j, where solution i is in APF. The numbers in the bottom row of Table II show that the average relative distances of solutions find by AMOSA and MOACS that are not in APF to solutions in APF is small.

Figure 4 shows the running time of AMOSA and MOACS in 20 simulated CRNs within 200 iterations and 1000 iterations, respectively. For all 20 CRNs, the running time of AMOSA and MOACS for 1000 iterations is within 5 seconds, and even less when running them for 200 iterations. Moreover, for most cases, the running time of AMOSA is 1 to 2 seconds longer than the running time of MOACS.

From the results above we can see that AMOSA and MOACS can find most solutions that are within or close to the APF in only a few seconds. However, one advantage of MOACS is that, unlike AMOSA, it is not limited by the annealing temperature, and it can be run continuously, hence adapting to the dynamic nature of CRNs.

VI. CONCLUSIONS

In this paper, we proposed a new modeling method to model a CRN into a MMHG. Using this MMHG, we proposed an approach for building multicast trees and studied scheduling algorithms to compute the $delay$ and $rate$ of a multicast tree. We applied two metaheuristic algorithms to find the Pareto Front of multicast routing trees with multiple objectives. Our simulation results show that in small CRNs, MOACS can find over 60% of the APF within only a few seconds, and AMOSA can find approximately 45%. The solutions found that are not in the APF are within 10% relative distances from the solutions in APF. Moreover, MOACS performs better than AMOSA as the speed of AMOSA in finding optimal solutions is limited by the annealing temperature.

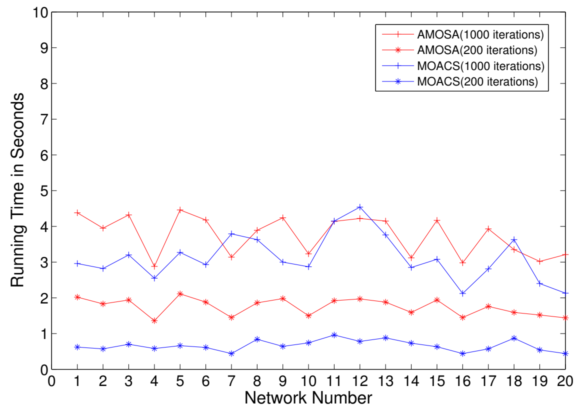

Figure 4. Running time comparison between AMOSA and MOACS

REFERENCES

[1] D. Pinto, and B. Barán. "Solving multiobjective multicast routing problem with a new ant colony optimization approach." Proceedings of ACM, international IFIP/ACM Latin American conference on Networking, 2005.
[2] S. Bandyopadhyay, et al. "A simulated annealing-based multiobjective optimization algorithm: AMOSA." Evolutionary Computation, IEEE Transactions on 12.3 (2008): 269-283.
[3] J. Jin, H. Xu, and B. Li. "Multicast scheduling with cooperation and network coding in cognitive radio networks." INFOCOM, IEEE, 2010.
[4] D. T. Ngo, C. Tellambura, and H. H. Nguyen. "Resource allocation for OFDM-based cognitive radio multicast networks." WCNC, IEEE 2009.
[5] H. M. Almasaeid, and A. E. Kamal. "Exploiting Multichannel Diversity for Cooperative Multicast in Cognitive Radio Mesh Networks." IEEE/ACM Transactions on Networking, 2014.
[6] C. Gao, et al. "Multicast communications in multi-hop cognitive radio networks." Selected Areas in Communications, IEEE Journal on 29.4 (2011): 784-793.
[7] W. Kim, et al. "CoCast: multicast mobile ad hoc networks using cognitive radio." MILCOM, IEEE, 2009.
[8] White spaces (radio), webpage:http://en.wikipedia.org/wiki/White_spaces _(radio)
[9] S. H. Alnabelsi, and A. E. Kamal. "Resilient Multicast Routing in CRNs Using a Multilayer Hyper-graph Approach." ICC, IEEE, 2013.
[10] S. Krishnamurthy, et al. "Time-Efficient Layer-2 Auto-Configuration for Cognitive Radios." IASTED PDCS. 2005.
[11] K. Deb, et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II." Evolutionary Computation, IEEE Transactions on 6.2 (2002): 182-197.
[12] M. Laumanns. "SPEA2: Improving the strength Pareto evolutionary algorithm." Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems, page 95–100. Athens, Greece, International Center for Numerical Methods in Engineering, (2001).
[13] W. Wang, et al. "Efficient interference-aware TDMA link scheduling for static wireless networks." Proceedings of ACM, international conference on Mobile computing and networking, 2006.