Virtual Link Embedding in Software-Defined Multi-radio Multi-channel Multi-hop Wireless Networks

Lunde Chen, Slim Abdellatif, Armel Francklin Simo Tegueu, Thierry Gayraud

Virtual Link Embedding in Software-Defined Multi-radio Multi-channel Multi-hop Wireless Networks

Lunde Chen
Slim Abdellatif
Armel Francklin Simo
CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
Univ de Toulouse, INSA, LAAS, F-31400 Toulouse, France

Thierry Gayraud
CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
Univ de Toulouse, UPS, LAAS, F-31400 Toulouse, France

ABSTRACT

There is rising interest in applying SDN principles to wireless multi-hop networks, as this paves the way towards bringing the programmability and flexibility that is lacking in today's distributed wireless networks (ad-hoc, mesh or sensor networks) with the promising perspectives of better mitigating issues as scalability, mobility and interference management and supporting improved controlled QoS services.

This paper investigates this latter aspect and proposes an Integer Linear Programming (ILP) based wireless resource allocation scheme for the provision of point-to-point and point-to-multipoint end-to-end virtual links with bandwidth requirements in software-defined multi-radio multi-channel wireless multi-hop networks. The proposed scheme considers the specificities of wireless communications: the broadcast nature of wireless links which can be leveraged for point-to-multipoint links resource allocations, and, the interference between surrounding wireless links. It also considers switching resource consumption of wireless nodes since, for the time being, the size of SDN forwarding tables remains quite limited. A Genetic Algorithm derived from the ILP formulation is also proposed to address the case of large wireless networks. Our simulation results show that both methods work effectively.

KEYWORDS

wireless SDN, resource allocation, Quality of Service, virtual link embedding, multicast, genetic algorithm.

1 INTRODUCTION

Applying Software Defined Networking (SDN) design principles to wireless networks can pave the way to the emergence of novel and effective wireless network control applications (routing, network resource allocation, mobility management, energy management, etc.) with diverse expected benefits [1], notably, an improved global network performance, end-to-end network services with enhanced Quality of Service (QoS), etc.

Indeed, under the assumption of an effective topology discovery service [2] that allows SDN controllers to build an updated global and comprehensive view of the network, network control algorithms can leverage on this global and detailed view to derive informed and wise control decisions that are able to accommodate with the dynamicity of the network and flows' QoS requirements. Moreover, the flow level forwarding capability of SDN allows unprecedented fine-grained control on the traffic that is flowing in the network. Some of the prominent works from the literature that attempt to apply SDN to wireless networks in order to dynamically control the traffic for an improved provided QoS are : [3], [4] and [5] respectively in the context of wireless ad-hoc, wireless sensor and wireless mesh networks.

The focus of this work is on the design of resource allocation methods that enable the on-demand provision of network services with QoS requirements in an SDN enabled multi-radio multi-channel multi-hop physical network. The network service is expressed as a set of point-to-point and point-to-multipoint unidirectional end-to-end virtual links (VLs), each with its own bandwidth requirement.

Two methods are proposed in this paper. Both aim at mapping the requested virtual links on the substrate wireless network by computing the data paths that minimize and balance link and switching resource consumption as well as interference between wireless links while satisfying the QoS requirements. They also consider and account for some of the specificities of wireless communications, namely the broadcast nature of wireless links, and the mutual interference caused by transmissions on neighboring links. An Integer Linear Programming based formulation method is proposed to compute the optimal allocations for small and moderate size networks as well as an accompanying genetic algorithm for large networks. A Performance Evaluation is conducted for both methods.

The paper is organized as follows. Section II reviews previous work from the literature that are related to virtual network resource allocation in wireless networks. Then, section III introduces the system model used in our formulations. Section IV describes the ILP formulation and Section V describes the genetic algorithm formulation. Section VI presents the performance analysis of the proposed methods. Finally, Section VII concludes the paper.

2 RELATED WORKS

Virtual network embedding has attracted lots of attention in recent years. While most embedding schemes are conceived for wired substrate networks, existing works also considered the case where the substrate network is a wireless one. Table 1 summarizes existing works in the field of virtual link resource allocation, classified according to the virtual link types, QoS support, considered node resources, etc. (Referring to table 1, VL stands for virtual link, P2P and P2MP for Point-to-Point and Point-to-Multi-Point, ILP for Integer Linear Programming and GA for Genetic Algorithm).

As shown in Table 1, to the best of our knowledge, our work stands out as the first to address the issue of virtual link embedding in software-defined multi-radio multi-channel multi-hop wireless

Table 1: Classification of virtual link resource allocation schemes for wireless multi-hop networks

	VL type	VL QoS	multi-radio	multi-channel	Network model specificity	Method	Node resources	Support path-split
[6]	P2P (Anypath)	packet loss delay	no	no	link metrics (EATT and EATX)	Incremental virtual network embedding	None	No
[7]	P2P	bandwidth	no	no	mobility	backtracking based heuristic	CPU, storage etc.	Not supported but discussed
[8]	P2P	bandwidth	no	no	interference matrix	heuristic	CPU	No
[9]	P2P	bandwidth	yes	yes	distance based interference model	greedy algorithm and GA	None	No
[10]	P2P	bandwidth	yes	yes	SINR-based interference model	ILP and heuristic	CPU	yes
our work	P2P P2MP	bandwidth	yes	yes	conflict graph based interference model	ILP and GA	flow table entries and group entries	Yes for ILP, No for GA

networks. Moreover, our work does not only consider point-to-point virtual links, but also addresses the case of point-to-multipoint virtual links. It also takes into account switching resources which is crucial especially when dealing with SDN based substrate networks.

3 SYSTEM MODEL

3.1 Network Model

Each node in a wireless multi-hop multi-radio multi-channel network is equipped with one or multiple Network Interface Cards (NICs). Each NIC is tuned to a channel and, any two NICs at the same node are tuned to different channels, in order to efficiently and fully make use of radio resources.

We assume that the channel assignment is given and static. There are in total $|\Lambda|$ non-overlapping frequency channels in the system and each node is equipped with q NICs where $q \leq |\Lambda|$. The channel capacity of λ is noted as B_λ.

We model the multi-hop multi-radio multi-channel network as a directed graph $G = (V, E)$ where V is the set of SDN nodes and $E \subseteq V \times V$ the set of bidirectional physical links which operate in half-duplex mode.

As we consider an OpenFlow-enabled infrastructure, to each node $v \in V$, is associated a switching capacity L_v, which is the maximum number of entries (i.e. size limit) of its flow table. The current size of node v flow table is denoted by L'_v.

An OpenFlow group table is also considered. A group table entry is either used to duplicate packets belonging to a point-to-multipoint virtual link on different network interfaces or to divide a flow of packets on many interfaces to implement path splitting. Similarly, M_v and M'_v denote respectively the maximum and current size of the group table of node v. We assume that we have already obtained a good channel assignment ζ. ζ assigns to each node $v \in V$ a set of $\zeta(v)$ of $|\Lambda|$ different channels : $\zeta(v) \subseteq \Lambda$.

A pair of NICs can communicate with each other if they are on the same channel and are within the transmission range of each other. In other words, the wireless link $e = ((u, v), \lambda)$, $u, v \in V$ and $\lambda \in |\Lambda|$ belongs to the substrate network, if channel $\lambda \in \zeta(u) \cap \zeta(v)$ and on this latter channel, nodes u and v are within the transmission range of each other. The set of neighbours of v (via any channel) is noted as $N(v)$.

3.2 Interference Model

Our interference model is based on the concept of conflict graphs [11]. A conflict graph is related to a channel. It describes the presence of interferences (represented as edges in the graph) between pairs of links (represented as vertices) if both links are active simultaneously. Following the logic of some previous works [12], from the conflict graph, we derive maximal cliques [13]. A clique is a subgraph of the conflict graph where all nodes interfere with each other. Maximal cliques are jointly or individually used in different ways to derive different kinds of constraints on the transmission rates of the wireless links that form the clique. For instance, since links belonging to a maximal clique cannot be active simultaneously, their aggregate transmission rate must be lower than the channel capacity.

There are different ways to build the conflict graphs with different levels of accuracy in capturing the interference. In this work, we do not promote any method and assume that we have the maximal conflict graphs as input. Thanks to the centralized nature of SDN and the adoption of an effective topology discovery service at the controller, the appropriate conflict graphs that meet the needs of network control applications can be made available by the controller. Also, we do not stick to any method to exploit the maximal cliques even if we have chosen one, for illustration, in our formulation of the resource allocation problem.

In the following, we denote the set of maximal cliques as C. We also denote the set of wireless links that form a clique c as E_c, and the nodes of the substrate network in the clique as S_c.

3.3 Virtual Links Request Model

A virtual links request consists of a set of $|K|$ virtual links. Each virtual link $k \in K$ is characterized by:

- a source node $s_k \in V$, and a set of destination nodes $T_k \in V \backslash \{s_k\}$ (when $|T_k| = 1$, the VL is point-to-point, otherwise it is point-to-multipoint);
- a bandwidth requirement of b_k;

The sequence of virtual links requests is noted as $\vec{\mathbf{K}} = [K_1, K_2, ...,]$.

4 ILP PROBLEM FORMULATION

Based on our previous work [14] whose focus was on wired SDN networks, this section describes our ILP formulation of the online virtual links resource allocation on an SDN based wireless multi-hop substrate network. In comparison to the previous work, this formulation adds in many aspects by taking into consideration (1) the broadcast nature of wireless links, which is used as a leverage to efficiently support point-to-multipoint virtual links, as well as, (2) the interferences between surrounding links which is minimized and distributed on different cliques (regions) to improve the admissibility of forthcoming virtual links requests. Below, the variables and problem constraints are listed. Then, the considered objective function is defined.

4.1 Resource-related assignment variables

The resource allocation algorithm provides as output the set of routes with the needed resources to support each of the virtual links (with its required QoS) that composes a request. As cited above, two types of network resources are considered : the bandwidth of wireless links and the switching resources (flow table and group table entries). Since VLs may be point-to-multipoint, it is likely that resource allocations will be mutualized close to the source and as we get closer to destinations, they will tend to be more dedicated to specific destinations. As a consequence, basic assignment variables are related to a specific destination of a VL. Our model considers the following variables:

- $f_k^t((v,u),\lambda)$ is an integer variable that represents the bandwidth allocated at link $((v,u),\lambda)$ to the packets of VL k that are flowing from the origin node s_k to a destination node t. More generally, $f_k((v,u),\lambda)$ refers to the amount of bandwidth used on link $((v,u),\lambda)$ by the VL k, whatever the destination. It is set to the maximum of $f_k^t((v,u),\lambda)$ for all $t \in T_k$. Specific to the broadcast nature of wireless medium, in which one node can deliver a paket to multiple neighbors from one transmission, we also introduce an integer variable denoted as $f_k(v,\lambda)$ that refers to the amount of bandwidth used on channel $\lambda \in \zeta(v)$ by node v to support the VL k. It is set to the maximum of $f_k((v,u),\lambda)$ for all $u \in N(v)$ such as $((v,u),\lambda) \in E$.
- $l_k(v)$ is a binary variable that indicates the number of flow table entries consumed by VL k at node v. An entry is installed in node v flow table if at least one of its adjacent physical links supports the VL. Formally:

$$\forall k \in K, \forall v \in V, \forall u \in N(v), \\ \forall \lambda \in \zeta(v) \cap \zeta(u) : g_k((v,u),\lambda) \leq l_k(v) \quad (1)$$

$$\forall k \in K, \forall v \in V, \forall u \in N(v) : \\ l_k(v) \leq \sum_{\lambda \in \zeta(v) \cap \zeta(u)} (g_k((v,u),\lambda) + g_k((u,v),\lambda)) \quad (2)$$

where $g_k((v,u),\lambda)$ is an intermediate binary variable that equals 1 if some bandwidth is assigned to VL k at link $((v,u),\lambda)$, 0 otherwise. It is derived from some other intermediate variables $g_k^t((v,u),\lambda)$ that, in turn, indicates whether some bandwidth is assigned to the flow of packets of VL k destined to $t \in T_k$ in link $((v,u),\lambda)$ (i.e. $g_k^t((v,u),\lambda) = 0$ if $f_k^t((v,u),\lambda) = 0$ and 1 otherwise).

- similarly, $m_k(v)$ is a binary variable indicating if a group table entry is assigned to VL k at node v. A group table entry is added when splitting a flow of packets belonging to k at node v or when duplicating packets (for point-to-multipoint VLs) on two or more links that operate on distinct channels. This is expressed as:

$$\forall k \in K, \forall v \in V : \\ m_k(v) = \begin{cases} 0 & if \ \sum_{\lambda \in \zeta(v)} g_k(v,\lambda) \leq 1 \\ 1 & otherwise \end{cases}$$

where $g_k(v,\lambda)$ is an intermediate boolean variable that indicates if node v relays packets from VL k on channel λ, whatever its neighbors on this channel. It is derived from the set of previous variables $g_k((v,u),\lambda)$ with $u \in N(v)$ and $\lambda \in \zeta(v) \cap \zeta(u)$. This equation could be easily linearized as follows:

$$\forall k \in K, \forall v \in V : \\ 2m_k(v) \leq \sum_{\lambda \in \zeta(v)} g_k(v,\lambda) \leq 1 + |\Lambda| \, m_k(v) \quad (3)$$

- ξ_{max} and ξ_{min} which refer to the maximum and minimum clique utilization after request acceptance (i.e. by taking into account the bandwidth allocations consumed by the virtual links that compose the request).
- l_{max} and l_{min} which similarly refer to the maximum and minimum flow table utilization (when considering all network nodes) after request acceptance.

4.2 Problem constraints

The constraints on bandwidth allocations are described hereafter in equations 4 to 12. The constraints related to switching resources allocation is given by inequalities 4 and 5. They simply ensure that the total number of flow and group table entries assigned to VLs composing the request, does not exceed available nodes' flow and group tables entries. Equation 6 reflects the linearization of the *Max* and *Min* operator applied to the variables $l_k(v)$ to get l_{max} and l_{min}.

$$\forall v \in V : \sum_{k \in K} l_k(v) \leq L_v - L'_v \quad (4)$$

$$\forall v \in V : \sum_{k \in K} m_k(v) \leq M_v - M'_v \quad (5)$$

$$\forall v \in V : l_{min} \leq L_v - L'_v + \sum_{k \in K} l_k(v) \leq l_{max} \quad (6)$$

Constraint 7 reflects the linearization of the maximum bandwidth $f_k^t(e)$ allocated to VL k at link $e = ((v,u),\lambda)$, whatever the destination. Equation 8 ensures that the total bandwidth assigned to the

substrate wireless nodes that belong to the clique does not exceed the remaining bandwidth of the clique. In this equation, each maximal clique with it's associated channel λ is noted as $(c, \lambda) \in C$, which is composed of E_c, and the residual capacity is $\xi(c) = B_\lambda - \xi'(c)$, with $\xi'(c)$ denoting the bandwidth allocations related to already admitted virtual links on all the physical links that compose the clique c. Equation 9 reflects the linearization of the *Max* and *Min* operator applied to the variables $f_k(v, \lambda)$ to get ξ_{max} and ξ_{min}. Equation 10 presents the usual flow conservation constraints.

$$\forall k \in K, \forall e = ((v,u),\lambda) \in E, \forall t \in T_k : f_k^t(e) \leq f_k(e) \tag{7}$$

$$\forall (c,\lambda) \in C : \sum_{k \in K} \sum_{v \in S(c)} f_k(v,\lambda) \leq B_\lambda - \xi'(c) \tag{8}$$

$$\forall (c,\lambda) \in C : \xi_{min} \leq B_\lambda - \xi'(c) + \sum_{k \in K} \sum_{v \in S(c)} f_k(v,\lambda) \leq \xi_{max} \tag{9}$$

$$\forall k \in K, \forall t \in T_k, \forall v \in V :$$

$$\sum_{u \in N(v)} \sum_{\substack{\lambda \in \zeta(u) \cap \zeta(v) \\ e_1((v,u),\lambda) \\ e_2((u,v),\lambda)}} (f_k^t(e_1) - f_k^t(e_2)) = \begin{cases} b_k & if\ v = s_k \\ -b_k & if\ v = t \\ 0 & else \end{cases} \tag{10}$$

Equation 11 is a channeling constraint between integer and binary variables: $f_k((v,u),\lambda)$ and $g_k((v,u),\lambda)$. It also constrains the VL k's bandwidth assignment at a physical link to the requested bandwidth b_k. Equation 12 constrains the bandwidth that is assigned to the flow of packets destined to a specific VL's end-point (or destination) within a range of values, in addition to establishing a channeling constraints between binary and integer variables. The inequality on the right side ensures that the bandwidth requirement of the VL is never exceeded. The inequality on the left side directs path-splitting and avoids the multiplication of splits with low bandwidth allocations. Indeed, if active, path-splitting is feasible only if the bandwidth allocated to the splits respects a minimum threshold b_k^{min}. In practice, b_k^{min} is a ratio of b_k, $b_k^{min} = PS_{ratio} * b_k$ with $PS_{ratio} \in [0,1]$ (then, $PS_{ratio} \leq 0.5$ when the path-splitting is allowed, and $PS_{ratio} = 1.0$ when it is forbidden).

$$\forall k \in K, \forall v \in V, \forall u \in N(v), \forall \lambda \in \zeta(v) \cap \zeta(u) :$$
$$g_k((v,u),\lambda) \leq f_k((v,u),\lambda) \leq b_k * g_k((v,u),\lambda) \tag{11}$$

$$\forall k \in K, \forall t \in T_k, \forall v \in V, \forall u \in N(v), \forall \lambda \in \zeta(v) \cap \zeta(u) :$$
$$b_k^{min} * g_k^t((v,u),\lambda) \leq f_k^t((v,u),\lambda) \leq b_k * g_k^t((v,u),\lambda) \tag{12}$$

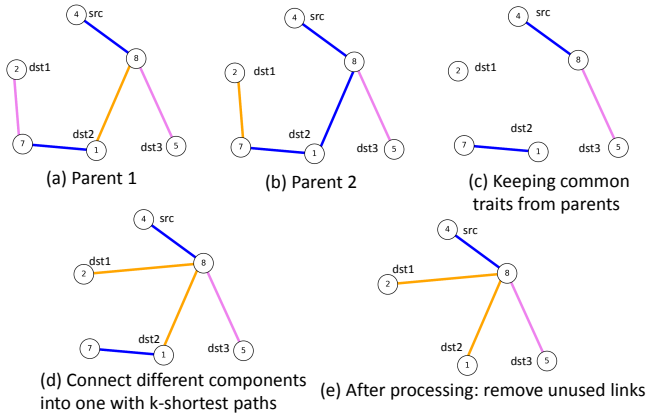

Figure 1: Crossover of two parent trees to get a offspring

4.3 Objective function

Minimize

$$\alpha_1 \sum_{k \in K} \sum_{v \in V} \sum_{\lambda \in \zeta(v)} f_k(v,\lambda) + \alpha_2 \sum_{k \in K} \sum_{v \in V} l_k(v)$$
$$+ \alpha_3 \sum_{k \in K} \sum_{v \in V} m_k(v) + \beta_1 \sum_{(c,\lambda) \in C} \sum_{v \in S(c)} \sum_{k \in K} |E(c)| f_k(v,\lambda)$$
$$+ \beta_2(\xi_{max} - \xi_{min}) + \beta_3(l_{max} - l_{min}) \tag{13}$$

The objective function is set to take into account both the resource consumption and the interference introduced by bandwidth allocations on surrounding links. For that, the main objective of our approach is to minimize the total resources required to map virtual links, which is represented by the first three terms that cover respectively links bandwidth, flow tables and group tables resources. In addition, the objective function mitigates the interference between links by avoiding overloading links belonging to cliques with a high number of members. This favors radio resource spatial reuse, increasing the overall available network resources. Finally, the last two terms aim at reducing the disparities of cliques' bandwidth utilization and flow tables' utilization. They also contribute improving flow admissibility.

5 GENETIC ALGORITHM

Exact solutions to the considered problem can be obtained by solving our previously presented ILP-based algorithm. However, the complexity of computation, which increases exponentially with the number of parameters (number of nodes, links, radios and channels etc.), might make it practically infeasible for large networks. Therefore, a practically feasible approach is to find a proficient near-optimal solution while sustaining realistic performance. In this section, we present a genetic algorithm based solution to address this aspect. The overall work flow of our GA scheme is described in Algorithm 1. Hereafter we present the detailed algorithm.

5.1 Encoding scheme

To use a GA, it is necessary to choose a representation that defines the genotype of an individual which is conceptually designated as a chromosome. In our case, an individual is a possible solution to a

Algorithm 1: GA-based Resource Allocation

Input : $G(V,E); K; W = [w_{e_1}, w_{e_2}, ..., w_{e_{|E|}}]$; $\alpha_1; \alpha_2; \alpha_3; \beta_1; \beta_2; \beta_3; N_p; N_g; cxPB; mutPB$;
Output: $\chi = [\tau_1, \tau_2, ..., \tau_{|K|}]$ (i.e. the best individual)

1 **begin**
2 $P_0 \longleftarrow$ `InitialPop`(G, K, N_p, W)
3 $P \longleftarrow P_0$
4 **for (** $j_g = 0;\ j_g < N_g;\ j_g++$ **)** {
5 **for (** $j_p = 0;\ j_p < N_p;\ j_p++$ **)** {
6 $(\chi^a, \chi^b) \longleftarrow$ `TournamentSelection`(P)
7 $\chi^c \longleftarrow$ `Crossover`$(G, K, (\chi^a, \chi^b), cxPB, W)$
8 $P \longleftarrow P \cup$ `Mutation`$(\chi^c, mutPB)$
9 $P \longleftarrow$ `PopulationSelection`$(P, N_p, \alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \beta_3)$
10 $\chi \longleftarrow$ `SelectBestIndividual`(P)

Algorithm 2: Initial Population Computation

Input : $G(V,E); K; N_p; W = [w_{e_1}, w_{e_2}, ..., w_{e_{|E|}}]$
Output: $P_0 = \{\chi^i, \forall i \in \{1, ..., N_p\}\ with\ \chi^i = [\tau_1^i, \tau_2^i, ..., \tau_{|K|}^i]\}$

1 **begin**
2 $P_0 \longleftarrow \emptyset$
3 $G'(V', E') \longleftarrow$ `Clone`$(G(V,E))$
4 $W' \longleftarrow$ `Clone`(W)
5 **for (** $i = 0;\ i < N_p;\ i++$ **)** {
6 **foreach** $e \in E'$ **do**
7 $W'[e] \longleftarrow W'[e] \times$ `Random`$(1, 1.5)$
8 **foreach** $k \in K$ **do**
9 $\tau_k^i \longleftarrow$ `ComputeSteinerTree`$(G', W'[e], s_k, T_k)$
10 $\chi^i[k] \longleftarrow \tau_k^i$
11 $P_0 \longleftarrow P_0 \cup \chi^i$

request for resource allocation. Recall that each request K consists of a set of point-to-point and/or point-to-multipoint virtual links. We naturally represent an individual i denoted by χi as a vector of genes where each gene τ_k maps resources assigned to a virtual link $k \in K$. In other words, $\chi i[k] = \tau_k^i$ refers to gene k of individual i. As our GA doesn't take into consideration the case of path splitting, a tree connecting the source s_k to the destination nodes $t \in T_k$, is sufficient to represent a gene. This tree is in fact a subgraph of the substrate network graph G. Each tree is associated with switching resources and links bandwidth respectively allocated at each substrate node and link belonging to this tree.

5.2 Initial population

The first step in the functioning of a GA is the generation of an initial population (Algorithm 1 - line 2). It is computed by generating a given population size (N_p) with each member of this population encoding an individual representing a possible solution. One important objective is to have a reasonable diversity among the initial population, in order to avoid premature local convergence. In our case, as detailed in Algorithm 2, to generate an individual i (Algorithm 2 - line 4), we compute the minimum Steiner tree as a routine to build the tree representation of each genes k (Algorithm 2 - line 9). Note that in the case of multiple links between two nodes, the link with the minimum link cost is sustained for Steiner tree construction. To bring diversity, at each Steiner tree construction, the cost of each link in the substrate network is multiplied by a random factor in the range of $[1, 1.5]$.

5.3 Fitness function

After creating the initial population, each individual is evaluated and assigned a fitness value according to a fitness function. The optimality of a solution is defined by its corresponding fitness value. Equation 14 defines our fitness function.

$$\begin{aligned} F(\chi) =& (F_{bw}(\chi) + F_{openflow}(\chi) + F_{group}(\chi) + F_{interf}(\chi) \\ &+ F_{bw\_balance}(\chi) + F_{sw\_balance}(\chi)) \\ &* \hat{F}_{cliques}(\chi) * \hat{F}_{sw}(\chi) \end{aligned} \tag{14}$$

where

$$F_{bw}(\chi) = \alpha_1 \sum_{\tau \in \chi} \sum_{e \in \tau} \phi(\tau, e) b_\tau$$

$$F_{openflow}(\chi) = \alpha_2 \sum_{\tau \in \chi} \sum_{v \in V} \sigma(\tau, v)$$

$$F_{group}(\chi) = \alpha_3 \sum_{\tau \in \chi} \sum_{v \in V} \eta(\tau, v)$$

$$F_{interf}(\chi) = \beta_1 \sum_{\tau \in \chi} \sum_{(c,\lambda) \in C} \sum_{v \in S(c) \cap S(\tau)} \left|S(c)\right| b_\tau$$

$$F_{bw\_balance}(\chi) = \beta_2 * (max\_bw(C, \chi) - min\_bw(C, \chi))$$

$$F_{sw\_balance}(\chi) = \beta_3 * (max\_sw(V, \chi) - min\_sw(V, \chi))$$

$$\hat{F}_{cliques}(\chi) = 1 + 100 \sum_{(c,\lambda) \in C} \delta(\chi, c)$$

$$\hat{F}_{sw}(\chi) = 1 + 100 \sum_{v \in V} \theta(\chi, v)$$

In the fitness function, $\phi(\tau, e)$, $\sigma(\tau, v)$, $\eta(\tau, v)$, $\delta(\chi, c)$ and $\theta(\chi, v)$ are all indicator functions that take value 0 or 1. $\phi(\tau, e)$ indicates if an edge e in a tree τ supporting a virtual link is transmitting or not. It takes value 1 for all edges in the tree τ except those who use the multicast advantage: in the latter case, $\phi(\tau, e)$ takes value 0. b_τ is the requested bandwidth of a virtual link, and corresponds to the b_k in the ILP formulation. Hence we have $F_{bw}(\chi)$ which is the sum of bandwidth consumed by transmitting links. In the same manner, $\sigma(\tau, v)$ indicates if a node v is included in the tree τ or not, hence consuming one OpenFlow table entry. $\eta(\tau, v)$ indicates if a node v serves as a multicast node or not, hence consuming one group table entry. $F_{interf}(\chi)$ reflects the total interference brought by the instantiated virtual links. For space reasons, detailed explanations of $F_{bw\_balance}(\chi)$ and $F_{sw\_balance}(\chi)$ are not given here. They correspond to the maximum minus minimum clique bandwidth consumption among all cliques and maximum minus minimum flow table entries consumption among all nodes and can also easily be calculated with $\phi(\tau, e)$ and $\sigma(\tau, v)$.

Those six fitness terms correspond to the objective function in the ILP formulation, i.e. they give the same results when the virtual link embeddings are the same.

Unlike the ILP formulation, the constraints on cliques and switching resources are also included in the fitness function of GA, i.e. the $F_{cliques}(\chi)$ and $F_{sw}(\chi)$ multiplier. In a feasible solution, those two terms should be of value 1. However, in some cases due to the sparsity of feasible solutions, those infeasible solutions should not be removed from the population. In fact, some solutions are more infeasible than others, and should be reflected in our fitness function. To reflect to which extent a solution χ is far from a feasible solution, we penalize those infeasible solutions according to how seriously they violate the clique bandwidth and the switching resource constraints. $\delta(\chi, c)$ indicates if the bandwidth allocations chosen in χ respect clique c bandwidth constraint, i.e. the total bandwidth of transmitting links in c which come from χ, doesn't violate the constraint delimited by the minimum remaining capacity of all links in c. The 100 here is a large number (compared to 1 as a multiplier) that penalizes violations of constraints. The more we have clique constraint violations for χ, the larger the fitness function $\hat{F}_{cliques}(\chi)$. In the same manner, $\theta(\chi, v)$ indicates if a node respects its switching resource constraint or not, and $\hat{F}_{sw}(\chi)$ reflects to which extent the violation is serious, i.e. the number of nodes that doesn't respect its switching resource constraint.

5.4 Selection of parents and crossover scheme

In this stage (Algorithm 1 - Line 6), chromosomes from a population are selected for reproduction (crossover), detailed in Algorithm 3. The operation of selection aims at favoring reproduction and survival of the fittest individuals. We use tournament selection of size 3 to select a pair of chromosomes as the parents to produce an offspring by applying crossover operator between them, with the crossover probability *cxPB*. Its strategy is summarized in Algorithm 3. The idea is simple and consists to pass common traits from parents to offspring according to a specific logic called *Similitude* (Algorithm 3 - Line 4). In order to explain how this primitive works, let $\chi^a = [\tau_1^a, \tau_2^a, ..., \tau_K^a]$ and $\chi^b = [\tau_1^b, \tau_2^b, ..., \tau_K^b]$ be the selected parents. The crossover operator generates a child $\chi^c = [\tau_1^c, \tau_2^c, ..., \tau_K^c]$ by identifying the same links between τ_k^a and τ_k^b for each $k \in K$, and retaining these common links in τ_k^c (as in [15]). According to the definition of the fitness function, the "better" individual has higher probability of being selected as a parent and survive. Thus, the common links between two parents are more likely to represent the "good" traits. However, retaining these common links in τ_k^c may generate some separate sub-trees. Therefore, links are needed to be selected to connect these disconnected sub-trees into a tree. The whole process is illustrated in Figure 1. Moreover, to maintain diversity among solutions, instead of connecting separated components each time with the shortest path, we adopt a random k-shortest path (with k $\leq$ 3). Note that in the case of multiple links between two nodes, the link with the minimum link cost is used for k-shortest path construction. Finally, a post-processing can be required to remove isolated branches of the tree that contain neither the source node nor destination nodes. As the cross-over is carried out from one VL to a next one that compose the request, the function *updateProhibitiveLinkCost* (Algorithm 3 - Line 10) assigns an infinity link cost to links that belong to cliques with no longer bandwidth left for future VLs, and to links with one end node with no flow table entries left. In this way, infeasible solutions are kept away when possible.

Algorithm 3: Crossover Scheme

Input : $G(V, E)$; K; χ^a; χ^b; $cxPB$; $W = [w_{e_1}, w_{e_2}, ..., w_{e_{|E|}}]$;
Output: $\chi^c = [\tau_1^c, \tau_2^c, ..., \tau_{|K|}^c]$

1 **begin**
2 $G'(V', E') \longleftarrow$ `Clone`$(G(V, E))$
3 $W' \longleftarrow$ `Clone`(W)
4 **if** (`Random`$(0, 1) < cxPB$) **then**
5 **foreach** $k \in K$ **do**
6 $\tau_k^c \longleftarrow$ `Similitude`(τ_k^a, τ_k^b)
7 **while** `isNotConnected`(τ_k^c) **do**
8 $\tau_k^c \longleftarrow$ `randomKShortestPath`(G', W', τ_k^c)
9 $\chi^c[k] \longleftarrow \tau_k^c$
10 `updateProhibitiveLinkCost`(G', W', τ_k^c)

5.5 Mutation schemes

When a new offspring is produced, the mutation operation is performed according to the mutation probability *mutPB* (Algorithm 1 - Line 8). We identify two types of mutations that could be very helpful, i.e. (1) mutation based on link breaking and reconnection, (2) mutation based on channel transition. The action of mutation gives more chances of getting rid of local sub-optimal solutions. For the first type of mutation, the procedure randomly selects a link in the tree and remove it to create two separate sub-trees; then, it re-connects these separate sub-trees using a random k-shortest path. The second type of mutation come from the importance of channels in our problem. That is, when a selected link to mutate in the tree corresponds to a multi-link in the substrate network, it's possible to directly change the channel of this link. This could lead us to finding more opportunities of multicast advantage, or a better load balancing among cliques.

5.6 Balanced resource allocation with dynamic link cost

We set the normal link cost of $e = ((v, u), \lambda)$ as $w_e = \alpha_1 + \alpha_2 + \beta_1 \left|E(c)\right|$. If this static manner of defining the link cost is used across all requests in $\vec{\mathbf{K}} = [K_1, K_2, ...]$, cliques with low link costs are always favored in comparison to cliques with high link costs, regardless of their current load, leading to unbalanced cliques utilizations. Although in our fitness function the clique utilization balancing is taken into consideration, it's inefficient if most candidate solutions in GA lead to an unbalanced situation. To mitigate this, we give a dynamic version of link cost, as shown in Algorithm 4. The idea of the algorithm is that if the clique utilization of a clique c is among the top N_{top} most loaded, the link cost of links that form c should be multiplied by a factor of 1.1. If the clique c is

Algorithm 4: Balanced Resource Allocation With Dynamic Link Cost

```
1  Input  : G(V,E); K⃗; C; N_top; α1, α2, α3, β1, β2, β3,
            N_p, N_g, cxPB, mutPB
2  begin
3      foreach e ∈ E do
4          W[e] ⟵ α1 + α2 + β1 |E(c)|
5      foreach K ∈ K⃗ do
6          χ^K ⟵ geneticAlgorithm(G, K, W, α1,
             α2, α3, β1, β2, β3, N_p, N_g, cxPB, mutPB)
7          C_most ⟵ mostUsedCliques(C, N_top)
8          C_least ⟵ leastUsedCliques(C, N_top)
9          C_normal ⟵ C − C_most − C_least
10         foreach c ∈ C_most do
11             if c ∩ χ^K ≠ ∅ then
12                 foreach e ∈ c do
13                     W[e] ⟵ W[e] × 1.5
14             else
15                 foreach e ∈ c do
16                     W[e] ⟵ W[e] × 1.1
17         foreach c ∈ C_least do
18             if c ∩ χ^K = ∅ then
19                 foreach e ∈ c do
20                     W[e] ⟵ W[e] ÷ 1.5
21             else
22                 foreach e ∈ c do
23                     W[e] ⟵ W[e] ÷ 1.1
24         foreach c ∈ C_normal do
25             foreach e ∈ c do
26                 W[e] ⟵ α1 + α2 + β1 |E(c)|
```

yet being used in the current embedding of K (i.e. $\chi^K \cap c \neq \emptyset$, as shown in Line-11 of Algorithm 4), then an even higher multiplying factor (i.e. 1.5) should be given to links in c, before calculating the embedding solution of K_{next}. In this way, those most used cliques will be unfavored in the embedding of forthcoming requests, due to their high link costs. Note that the increase of link cost can be accumulated over time, i.e. if a clique stays always among the top most loaded from K_i to $K_{i+\Delta}$, its links costs will be increased $\Delta + 1$ times, until the clique is removed from the top most loaded list, at which point the normal link cost is given to the links in the clique. Inverse actions are carried out on the N_{top} least used cliques. At each iteration, links in other cliques are given normal link cost.

6 PERFORMANCE EVALUATION

The objectives of this performance analysis is to show that our methods clearly succeed in capturing three essential aspects of wireless links : (1) their broadcast nature which should be exploited whenever possible when embedding point-to-multipoint virtual links; and (2) interference between neighboring links which should be avoided whenever possible; and (3) to achieve a decent load-balancing of clique and flow table utilization to improve admissibility. It also compares both proposed methods and investigates the trade-off raised by these latter: accuracy versus computation time. Below, we describe our simulation model, the main performance metrics and some of the obtained results.

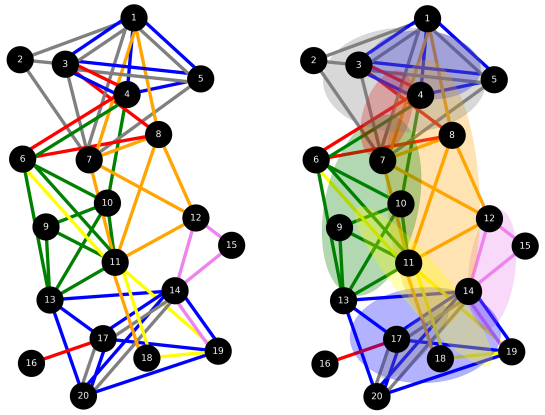

Figure 2: Network model used in our performance evaluation. Cliques are directly drawn on the illustrating graph, with ellipsoids covering the midpoint of each link in the clique.

6.1 Network Model

For space reasons, one single network instance is considered in the presented results. It is composed of 20 nodes connected via 60 links. Nodes are equipped with up to 3 radio interfaces that operate on 6 disjoint frequency bands (channels). The capacity of each channel is set to 180 units of bandwidth (UB). The left side of Figure 2 depicts the network topology, each link color reflects a frequency band. It leads to 9 cliques (as depicted in right side of Figure 2) with a number of members ranging from 3 to 11 links. Unless specified, the flow table and group table maximum size are set to 1000 and 100, respectively.

6.2 Load Model

Virtual links requests are composed of a number of point-to-point and point-to-multipoint virtual links randomly chosen between 4 and 6. Each point-to-multipoint virtual link has a number of destinations randomly chosen between 2 and 6. The bandwidth requirement of each virtual link is also chosen randomly from 1 to 3 UB. Source and destination selection is performed on a random basis. The request arrivals follow a Poisson process with an arrival rate r of 0.01, 0.02, 0.03, 0.04, i.e. in average 1, 2, 3 or 4 requests each 100 units of time (UT). The request life-time conforms to an exponential distribution with an average of 1000 UT.

6.3 Simulation Settings

The Integer Linear model was implemented in Python with CPLEX 12.63 solver. The experiments were carried out on a virtual machine with 25 vCPU and 16GB of RAM and running Ubuntu 14.04. A gap

of less than 1% to the optimal solution is considered satisfactory. Unless specified, path splitting is disabled for ILP. For GA, the implementation is in Python (running on pypy [1]) using deap [16]. Unless specified, population size is set to 18 and number of generations at 18. Cross-over probability is 0.9 and mutation probability is 0.05. N_{top} is set to 2. The simulation horizon is fixed to 10000 UT (this time period is sufficient to have our methods in the stationary regime). α_1, α_2 and α_3 are set to 1, 1 and 5 respectively throughout all evaluation experiments.

6.4 Performance metrics

The following performance metrics are computed during simulation for performance analysis purposes:

- Acceptance rate (*ac*, in %): the percentage of successful virtual links requests out of all the requests that arrived during the simulation time or accumulatively with the time.
- Clique utilization (*cu*, in %): bandwidth allocated at the links composing a clique divided by channel capacity.
- Switch resource utilization regarding flow table utilization (*su*, in %): flow table utilization at the nodes divided by the initial flow table size.
- Switch resource utilization regarding group table utilization (*gu*, in %): group table utilization at the nodes divided by the initial group table size.
- Computation time (in second): the average computation time for one request.

6.5 Performance Results

6.5.1 Coping with wireless links interference. The objective is to assess how efficient are our methods in reducing and avoiding wireless links interference. To this end, β_2 and β_3 are set to 0 in a first place. We compare the effect of setting β_1 to 1 versus to 0.

In fact, when embedding virtual links requests, our methods favor links belonging to cliques with limited number of members, introducing, by the way, in their surroundings less interference and, hence, preserving the overall available bandwidth. This is clearly shown in Figure 3 which focuses on the clique utilization of two groups of cliques: small cliques with a small number ($3 \sim 6$) of interfering links and large cliques with a high number ($8 \sim 11$) of links. When activating interference reduction, the bandwidth consumed by large cliques is decreased contrary to small cliques. As expected a portion of the bandwidth consumed by a large-size clique is transferred to smaller-size cliques: small cliques experience an increase in clique utilization while large cliques get less loaded.

Favoring small-size cliques may lead to longer data paths and hence more resources are needed to support the virtual links being embedded. Since the acceptance rate is improved with such a strategy, this means that this latter increase is compensated by the resource that are preserved thanks to interference reduction. With the considered network and load models, our experiments show a slight increase around 1% on the average length of selected data paths.

6.5.2 Assessing the gain brought by the Multicast advantage. The objective is to quantify the gain in resource usage that our methods

[1] http://pypy.org/

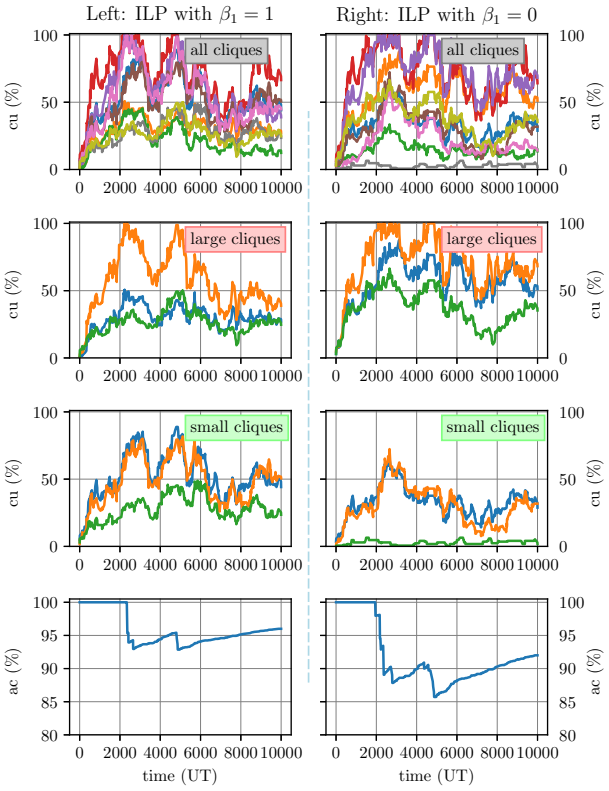

Figure 3: Clique utilization (all cliques, large cliques and small cliques) and acceptance rate with $\beta_1 = 1$ v.s $\beta_1 = 0$. Computed with ILP. Arrival rate = 0.02. $\beta_2 = 0$, $\beta_3 = 15$. Similar results are obtained with GA.

achieve by exploiting the multicast advantage when embedding point-to-multipoint virtual links. Again, β_2 and β_3 are set to 0 in a first place. β_1 is set to 1 as we have shown that interference should be taken into consideration for the modeling.

The clique utilization of the 9 cliques is presented in Figure 4. We see that disabling the multicast advantage induces extra bandwidth consumption that overloads all cliques and causes significantly more embedding failures.

6.5.3 Clique utilization balancing. By setting β_2 to 5, we activate clique load balancing. To show the effect of clique load balancing, Figure 5 shows that the clique utilizations have now much less disparity, with ILP as well as GA, compared to Figure 3 and 4 where $\beta_2 = 0$, leading to an improved acceptance rate (99.5% for ILP and 98.5% for GA) .

6.5.4 Switch resource consumption and balancing. To show how flow table resource is consumed and balanced and in which manner this might impact , we set the initial flow table size to 90, and compared the results of $\beta_3 = 0$ versus $\beta_3 = 15$ using ILP, as is shown in Figure 6. Apart from a much better balancing of flow table resource utilization, we also see an improved acceptance rate (99.5% v.s. 94.5%). Hence, flow table resource balancing should be activated. The effect of switch resource balancing of GA is shown

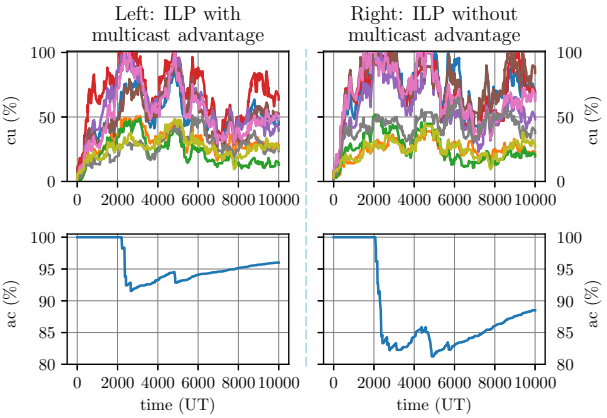

Figure 4: Clique utilization and acceptance rate with and without multicast advantage. Computed with ILP. Arrival rate = 0.02. $\beta_2 = 0$, $\beta_3 = 15$. Similar results are obtained with GA.

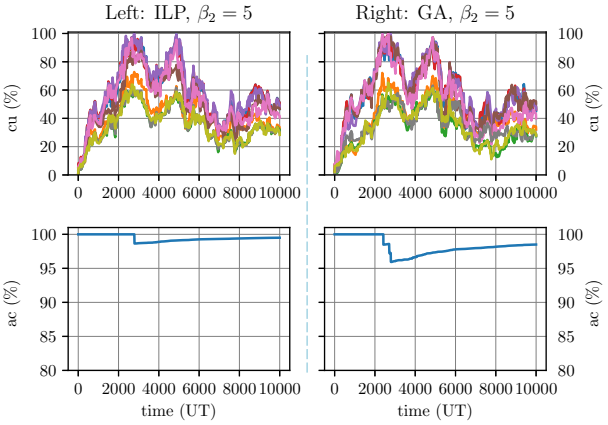

Figure 5: Clique utilization balancing with ILP and GA. Arrival rate = 0.02. $\beta_3 = 15$.

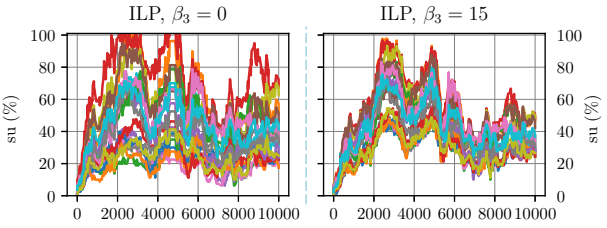

Figure 6: Switch resource consumption of all nodes, computed with ILP, with initial flow table size set at 90. Arrival rate = 0.02. $\beta_2 = 5$

in Figure 7. We can see that GA is less effective than ILP in switch resource balancing.

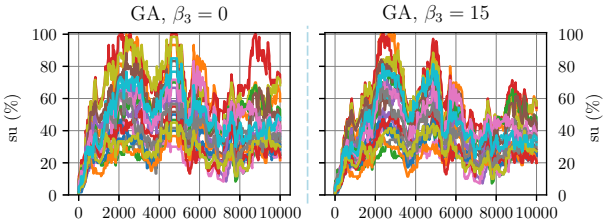

Figure 7: Switch resource consumption of all nodes, computed with GA, with initial flow table size set at 90. Arrival rate = 0.02. $\beta_2 = 5$.

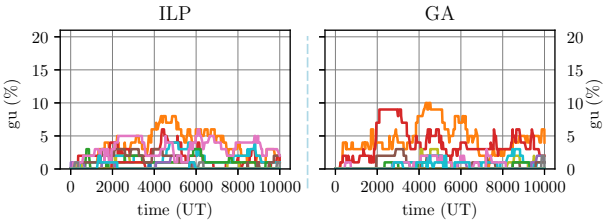

Figure 8: Group table consumption of ILP and GA. Arrival rate = 0.02. $\beta_2 = 5$, $\beta_3 = 15$.

Figure 8 presents the group table utilization, computed with ILP and GA. We observe that with ILP and GA, thanks to the multicast advantage, the group table consumption remains very limited despite the successful mapping of point-to-multipoint virtual links. As expected, for the considered simulation model, group table entries are abundant in comparison to embedding needs. As a consequence, it does not play a decent role in the selection of the data paths. Hence, there is no need to balance its utilization in the formulation.

6.6 Embedding method selection : ILP v.s. GA

There are two important criteria to consider when choosing the method to be applied: (1) accuracy (leading to optimal acceptance rate) and (2) computation time. Figure 9 shows the acceptance rate using different methods, and we can see that ILP-PS (ILP with path splitting enabled) gives slightly better results than ILP and GA. Figure 10 and Figure 11 present the acceptance rate and the computation time obtained with GA when considering different population and generation sizes. For the considered network model, GA with a population of 18 individuals and 18 generations lasts 80% of the computation time of ILP. With smaller population and generation size (e.g. 12 and 12), the computation time can be significantly reduced (less than 1/10 of ILP), bringing only minor degradation of the acceptance rate (~1.5%). Our experiments show that for larger network models, GA shows a significant advantage in computation time compared to ILP. On the contrary, with ILP-PS, as the search space explodes, the computation time can be several folds of that of ILP and hence much more than GA.

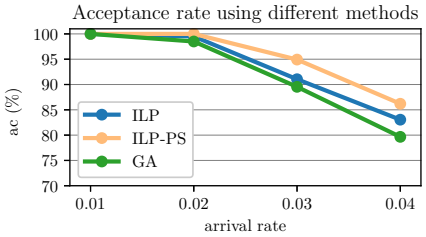

Figure 9: Acceptance rate with ILP, GA and ILP-PS, for different arrival rates. $\beta_2 = 5$, $\beta_3 = 15$.

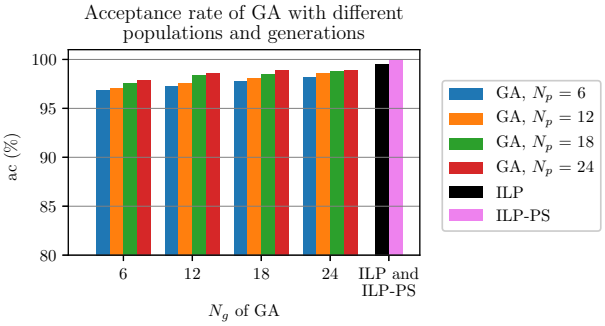

Figure 10: Acceptance rate of GA by varying number of generation and size of population, compared with ILP and ILP-PS. Arrival rate = 0.02

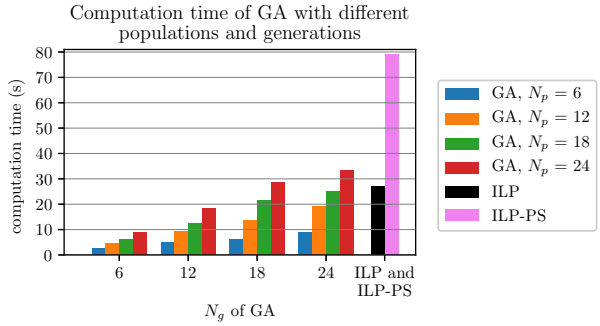

Figure 11: Average Computation Time of each request of GA by varying number of generation and size of population, compared with ILP and ILP-PS.

7 CONCLUSION

In this paper, we developed an Integer-Linear programming method and a genetic algorithm method for the resource allocation of multiple virtual links in wireless software defined multi-radio multi-channel multi-hop networks. In comparison to existing works, the main contribution or our proposals lies in the conjunction of the following features: (1) the support of point-to-multipoint virtual links in addition to point-to-point virtual links, and, in a wireless context, how to benefit from the multicast advantage to gain in bandwidth consumption, (2) the consideration of switching resources in the allocation of resources in addition to the bandwidth of channels. Through our evaluations, we show that both of our proposed methods work well. More interestingly, we investigated how the consideration of interference, multicast advantage as well as resource balancing could impact the embedding results, and, how the two proposed methods differ in performance and computation time.

ACKNOWLEDGMENT

This work was partially funded by the French National Research Agency (ANR) and the French Defense Agency (DGA) under the project ANR DGA ADN (ANR-13-ASTR- 0024) and by European Union's Horizon 2020 research and innovation programme under the ENDEAVOUR project (grant agreement 644960).

REFERENCES

[1] I. T. Haque and N. Abu-Ghazaleh,"Wireless Software Defined Networking: A Survey and Taxonomy," in *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2713-2737, Fourthquarter 2016.

[2] Lunde Chen, Slim Abdellatif, Pascal Berthou, Kokouvi Benoit Nougnanke, and Thierry Gayraud. "A Generic and Configurable Topology Discovery Service for Software Defined Wireless Multi-Hop Network". In *Proceedings of the 15th ACM International Symposium on Mobility Management and Wireless Access (MobiWac '17)*, 2017

[3] H. C. Yu, G. Quer and R. R. Rao, "Wireless SDN mobile ad hoc network: From theory to practice," *IEEE International Conference on Communications (ICC)*, Paris, 2017, pp. 1-7.

[4] A. De Gante, M. Aslan and A. Matrawy, "Smart wireless sensor network management based on software-defined networking," *27th Biennial Symposium on Communications (QBSC)*, Kingston, ON, 2014, pp. 71-75.

[5] Won Jin Lee, Jung Wan Shin, Hwi Young Lee and Min Young Chung, "Testbed implementation for routing WLAN traffic in software defined wireless mesh network," *International Conference on Ubiquitous and Future Networks (ICUFN)*, Vienna, 2016, pp. 1052-1055.

[6] M. Li, C. Hua, C. Chen and X. Guan, "Application-driven virtual network embedding for industrial wireless sensor networks," *IEEE International Conference on Communications (ICC)*, Paris, 2017, pp. 1-6.

[7] S. Abdelwahab, B. Hamdaoui, M. Guizani and T. Znati, "Efficient Virtual Network Embedding With Backtrack Avoidance for Dynamic Wireless Networks," in *IEEE Transactions on Wireless Communications*, vol. 15, no. 4, pp. 2669-2683, April 2016.

[8] Donggyu Yun, et al., "Embedding of virtual network requests over static wireless multihop networks", *Computer Networks*, Volume 57, Issue 5, 2013, pp 1139-1152

[9] Pin Lv, Xudong Wang, and Ming Xu. "Virtual access network embedding in wireless mesh networks". *Ad Hoc Netw.* 10, 7 (September 2012), 1362-1378.

[10] G. Di Stasi, S. Avallone and R. Canonico, "Virtual network embedding in wireless mesh networks through reconfiguration of channels," *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Lyon, 2013, pp. 537-544.

[11] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of Interference on Multi-hop Wireless Network Performance," *ACM Mobicom*, San Diego, CA, USA, September 2003

[12] J. Musacchio, R. Gupta, J. Walrand. "Sufficient Rate Constraints for QoS Flows in Ad-Hoc Networks". *INFOCOM*, 2005

[13] F. Cazals, C. Karande, "A note on the problem of reporting maximal cliques", *Theoretical Computer Science*, Volume 407, Issues 1-3, 6 November 2008, Pages 564-568

[14] M. Capelle, S. Abdellatif, M. J. Huguet and P. Berthou, "Online virtual links resource allocation in Software-Defined Networks," *IFIP Networking Conference (IFIP Networking)*, Toulouse, 2015, pp. 1-9.

[15] T. Lu and J. Zhu, "Genetic Algorithm for Energy-Efficient QoS Multicast Routing," in *IEEE Communications Letters*, vol. 17, no. 1, pp. 31-34, January 2013.

[16] François-Michel De Rainville, Félix-Antoine Fortin et al., "DEAP: a python framework for evolutionary algorithms". In *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation (GECCO '12)*, 2012